THE UNIVERSITY OF ALBERTA

Intersection and Minimum Distance Problems for Planar Polygons

by

Cao An Wang

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

Department of Computing Science

EDMONTON, ALBERTA

Fall 1983

Abstract

Most of the recent results in computational geometry rely on the attribute of convexity to provide greater efficiency. Given an arbitrary non-convex polygon, one approach is to decompose the polygon into convex pieces, then to handle each piece separately. On the other hand, the monotonic property of a polygon is also frequently used for further improvements.

In this thesis, several fundamental problems in computational geometry whose solutions are based on these two properties, convexity and monotonicity, are studied. The problems can be listed as follows:

1. polygon intersection reporting;
2. polygon intersection detection;
3. finding the minimum distance between two polygons in the L_1 and L_2 metrics.

The major results of the thesis are sumarized below.

1. $O(n)$ optimal algorithms to detect the intersection and to find the minimum distance between a convex n-gon (a polygon with n vertices) and a simple non-convex n-gon.
2. $O(\log n)$ optimal algorithms to find the minimum distance between two separable convex n-gons.
3. $O(n)$ optimal algorithms to find the minimum *vertex* distance between two separable convex n-gons.
4. An $O(m \log n)$ algorithm to report the intersection of a convex n-gon and a simple non-convex m-gon.
5. An $O(m \log n)$ algorithm to find the minimum vertex

distance between a convex n-gon and a simple non-convex m-gon, which are non-intersecting.

6. An $O(n \log n)$ algorithm to find the minimum distance between two separable simple non-convex polygons.

List of Figures

Table of Contents

Chapter 1

Introdution

1.1 The Nature of Computational Geometry

Geometry is an important branch of mathematics, which is used in almost all areas of modern science and technology. The increasing reliance on the computer for solving various scientific and technological problems forces computer scientists seriously to consider the relationship of pure traditional geometry (or non-algorithmic formulation) and algorithmic computational geometry. For example, a pure mathematician may never worry about the number of operations required to find the minimum distance between two planar polygons which contain thousands of vertices. But in the view of computer scientists, it is the number of operations that is an important factor for high speed computation. As a result, the task of translating the classical results into algorithmic terms is an important one for algorithm designers.

The pioneer in this field is Shamos, whose work constitutes the basis of computational geometry and provides a unified approach to solving many geometric problems such as intersection problems, inclusion problems, closest point problems, convex hull problems, etc. [Shamos 1975, Shamos and Hoey 1976]. Many computer scientists have given much attention not only to the fundamental problems of computational geometry itself, but also to its applications. Many areas involve fundamental computational geometry

problems. In graphics, for instance, essential operations such as windowing, clipping, positioning, etc. require the solution of hidden line or hidden face elimination problems, which involve computation of the intersection of two or more objects [Freeman and Shapira 1975]. In pattern recognition, emphasis is put on finding the convex hull of a set of points, the nearest neighbors of a point, and the diameter of an object [Toussaint 1980]. In robot moving problems, it is necessary to find the minimum distance of two objects and the diameter of an object [Shwartz 1981].

All the above applications are more or less related to the following three fundamental computational geometry problems.

(a) Polygon intersection reporting:

Given two polygons P and Q, report their common intersections, i.e., report $R(P) \cap R(Q)$, *where* $R(P)$ and $R(Q)$ represent the regions of polygons P and Q.

(b) Polygon intersection detection:

Given two polygons P and Q, determine whether they intersect each other, i.e., is $R(P) \cap R(Q) = \emptyset$.

(c) Minimum distance between two polygons:

Given two polygons, find the minimum distance[1] between them and report a pair of points which are separated by this distance.

---

[1]The minimum distance may be in the L_1 or L_2 metrics. Refer to pp. 45

1.2 Complexity Analysis

1.2.1 The Model of Computation

The model of computation used in this thesis is a random access machine (RAM) with real number arithmetic ability. We assume that in planar objects, each of the following operations can be done in constant time: finding the intersection of two line segments, computing the intersection angle of two line segments, finding the shortest distance between a point and a line segment, transforming one coordinate representation of a point to another, and determining on which side of a line a point lies.

1.2.2 Representation of a Polygon

In this thesis, only simple planar polygons are studied. [Shamos 1975]. A simple polygon is a polygon whose edges do not intersect themselves. The vertices of a simple polygon can be represented by ordered pairs, in either *Cartesian* coordinates $\langle x,y \rangle$ or *Polar* coordinates $\langle r,\theta \rangle$. $\theta(p)$, and $r(p)$ will denote the polar angle and the radius of a vertex p with respect to a specified origin. A polygon P of n vertices is called an n-gon, and is represented by a sequence of vertices $p_0, p_1, \ldots, p_n$. Any pair of two consecutive vertices (p_i, p_{i+1}), where the indices are modulo n, forms an *edge* of the polygon. A direction upon each edge is imposed such that the *interior* of the polygon $R(P)$ always lies to the righthand side of the edge. In other words, the edges around the boundary of P are

listed in clockwise order. In order to avoid confusion, P is used to denote the vertex chain, and $B(P)$ to denote the sequence of edges that forms the boundary of the polygon. From now on, tracing along the boundary of a polygon means tracing the boundary in clockwise order. For convenience sake, P is also used to denote a convex polygon, while Q denotes a non-convex simple polygon.

1.3 Outline of the Thesis

Efficient algorithms have been developed for some problems. The well known results for two simple n-gons can be summarized as follows.

(a) *Intersection reporting problem:*

two non-convex n-gons	$O((n+k)\log n)$[2]	[Bentley 1979]
two convex n-gons	$O(n)$	[Shamos 1975]

(b) *Intersection detection problem:*

two non-convex n-gons	$O(n\log n)$	[Shamos and Hoey 1976]
two convex n-gons	$O(\log n)$	[Chazelle 1980]

(c) *Minimum Euclidean distance problem:*

two convex n-gons	$O(\log^2 n)$	[Schwartz 1981]

---

[2] where k is the number of intersection points

In Chapter 2, an $O(m \log n)$ algorithm for reporting the common intersection of a convex n-gon and a non-convex m-gon is briefly outlined (intersection reporting problem). The algorithm is simpler than the general algorithm [Bentley 1979] for two non-convex polygons. An $O(n+m)$ optimal algorithm for determining whether a convex n-gon and a non-convex m-gon intersect each other is also presented (intersection detection problem).

In Chapter 3, several optimal algorithms for finding the minimum distance[3] between two separable convex n-gons, between two separable simple non-convex n-gons, and for determining the minimum distance between a convex n-gon and a non-convex m-gon, which do not intersect, are presented. Then the minimum vertex distance problem between two planar n-gons is discussed. An optimal $O(n)$ algorithm for finding the minimum vertex distance between two separable convex n-gons and a simple $O(m \log n)$ algorithm for finding the minimum vertex distance between a convex n-gon and a non-intersecting non-convex m-gon are presented. Finally, the minimum distance of two n-gons in the L_1 metric is studied.

In Chapter 4, the discussion is concluded and some open problems for further work proposed.

---

[3] in the L_2 (Euclidean) metric, unless specified otherwise.

Chapter 2

Intersection Reporting and Detection for Planar Polygons

In recent years planar geometric problems have attracted considerable attention. These problems arise in a number of applications including pattern recognition, hidden line elimination, etc. [Akl 1979, Avis 1981, Bentley 1979, Chazelle 1980, Dobkin 1976, Fisher 1980, Gindy 1981, Kirkpartrick 1982, Lee 1979, Muller 1977, Preparata 1977, Schachter 1978, Shamos 1975, 1976, Toussaint 1980, Snyder 1980]. In particular, the problems of detecting and finding the intersection of two planar polygons have been investigated by a number of researchers [Chazelle and Dobkin 1980, Shamos and Hoey 1976]. As stated in Chapter 2 these problems have been solved for the case of two convex polygons and detection of an intersection for two non-convex polygons. In the case of two convex polygons, the convexity property is used. In the case of two non-convex polygons the planar line sweep method is applied. Another solution for non-convex polygons is by decomposition into a set of convex parts, solving each part as a convex polygon. In this section, the case of a convex polygon and a non-convex polygon is considered; the solution is based on the monotonic and convex property. Formally, the above planar polygon problems can be represented as follows,

(a) Polygon intersection reporting problem:

Given a convex n-gon P and a non-convex m-gon Q, report their common intersections (i.e., report $R(P) \cap R(Q)$).

(b) Polygon intersection detection problem:

Given a convex n-gon P and a non-convex m-gon Q, determine whether or not an intersection occurs (i.e., $R(P) \cap R(Q) = \emptyset$?).

Here, two algorithms are presented (one of which is optimal). In Section 2.1, an $O(m \log n)$ algorithm is briefly outlined for reporting the common intersection of a convex n-gon and a non-convex m-gon (intersection reporting problem). In Section 2.2, an $O(n+m)$ optimal algorithm for determining whether a convex n-gon and a non-convex m-gon intersect is presented (intersection detection problem). Discussion of these results and conclusions are given in Section 2.3.

2.1 The Intersection Reporting Problem

For the problem of reporting the intersection of a convex n-gon and a non-convex m-gon, the planar sweeping algorithm, which is designed for two non-convex polygons, may be applied. That algorithm takes $O(m \log n)$ time [Bentley and Ottmann 1979]. Here, a different algorithm to solve the above problem is presented. Although this algorithm takes the same time order as the planar sweeping algorithm, it is somewhat simpler. The main idea of the improved method is to follow the boundary $B(Q)$ sequentially and find the intersection point(s) of $B(Q)$ with $B(P)$. In order to efficiently find the intersection points, *rays* for P are drawn from any point $o \in R(P)$. Each ray originates from o and passes through a

vertex of P, thereby partitioning the plane into n ordered *sectors* [Shamos 1975]. The sectors can then be ordered by their polar angles with respect to o and a binary search can be applied to the sectors to determine which sector contains the endpoints of each edge of $B(Q)$. Thus it can be determined whether or not an endpoint is in $R(P)$ and also the existence of an intersection of boundary $B(P)$ with a given edge e_i in $B(Q)$. If an intersection exists, then the edge e_j of $B(P)$ which intersects e_i can be found in $O(\log n)$ time.

Definition 1: Let S be an ordered set of objects, and ω be a mapping from S to another ordered domain. S is *monotonically increasing (decreasing)* with respect to ω iff for any $x,y \epsilon S$, $x<y$ implies $\omega(x) \leq \omega(y)$ $(\omega(x) \geq \omega(y))$. Let $x \epsilon S$, $S_{\geq}(x)=\{y \mid y \epsilon S$ and $y \geq x\}$ and $S_{\leq}(x)=\{y \mid y \epsilon S$ and $y \leq x\}$. S is *unimodal* with respect to ω iff for some $x'' \epsilon S$, $S_{\geq}(x'')$ and $S_{\leq}(x'')$ are non-empty, and $S_{\geq}(x'')$ is monotonically increasing (decreasing) and $S_{\leq}(x'')$ is monotonically decreasing (increasing) with respect to ω.

Example: Assume the points in boundary $B(P)$ are ordered such that $x<y$ if y is traced before x along P. Let P be a convex n-gon and $o \epsilon R(P)$ be the origin. P is monotonically decreasing with respect to a mapping θ, where $\theta(x)$ is the polar angle of x relative to o. Given a infinite line e, let d_e be a mapping which maps $x \epsilon B(P)$ to its shortest distance[4] from e. Then $B(P)$

---

[4]The distance is negative if x is on the other side of e.

is unimodal with respect to d_e.

It is easy to see that the intersection point(s) between the boundary $B(P)$ of a convex n-gon P and a line segment γ can be found in $O(\log n)$ time by making use of the unimodal property between $B(P)$ and the extended line γ and applying a binary search on the vertices of P.

Algorithm 1

The entire algorithm is described informally as follows. Let $P=(p_0,p_1,\ldots,p_{n-1})$, $Q=(q_0,q_1,\ldots,q_{m-1})$ be two polygons which can be represented by two linked lists. The boundary of Q, $B(Q)$, is traced sequentially and the intersection(s), if any, of an edge of $B(Q)$ with $B(P)$ is recorded. For each edge $e_i=(q_i,q_{i+1})$, one of the following four cases must occur.

a. If edge $e_i=(q_i,q_{i+1})$ is totally inside $R(P)$, (i.e., e_i is one of the edges bounding the intersection region), then continue with $e_{i+1}=(q_{i+1},q_{i+2})$.

b. If $q_{i+1}\epsilon R(P)$[5] and $q_i \notin R(P)$, then e_i intersects an edge of $B(P)$, say $h_j=(p_j,p_{j+1})$. Let k be that intersection point. Now create a new node k between p_j and p_{j+1}, and set up a special pointer in the linked list from k to q_{i+1}.

c. If $q_i\epsilon R(P)$ and $q_{i+1}\notin R(P)$, then set up a special

---

[5]This can be determined easily if the rays for P are drawn and it has been decided in which sector the endpoints of e_i lie.

pointer from q_i to the new node, say k' which has been inserted between p_j and p_{j+1}.

d. If both $q_i, q_{i+1} \notin R(P)$ and e_i does not intersect $B(P)$, then proceed directly to e_{i+1}. However, if e_i intersects $B(P)$ at two points, say k, k', then those two nodes are inserted into $B(P)$, as in the case of (b) and (c). Again a special pointer is set up from k to k'.

In order to report $R(P) \cap R(Q)$, the boundary $B(P)$ is traced until a node k with a special pointer is reached. Then follow this pointer and switch to $B(Q)$. $B(Q)$ is traced until a special pointer is reached, following that special pointer and switching to $B(P)$ again. In this manner, by switching between $B(P)$ and $B(Q)$, their edges can be traced. The area surrounded by the vertex chain loop forms one of the intersection regions to be reported. The above procedure is continued until all of the vertices of P have been examined. In order to avoid reporting a region more than once, all special pointers are removed after each has been traced.

As the algorithm references all m edges in $B(Q)$, n edges in $B(P)$, and determines whether each point of Q is in $R(P)$, the time complexity of the algorithm is $O(m \log n + n)$.

It is conjectured that the intersection reporting problem will take at least $O(m \log n)$ time and that the algorithm presented here is asymptotically optimal. The following theorem illustrates a lower bound for a similar problem.

Theorem 1: *Given a convex n-gon and a non-convex non-simple m-gon, the time complexity for the intersection reporting problem is at least $O(m \log n)$.*

Proof: Given a line and a point, assume the only operation is deciding on which side of the line the point lies. It is well known that deciding whether a point lies inside or outside of a convex n-gon P is at least $O(\log n)$. It can be shown by information theory that the same problem with m points is at least of $O(m \log n)$. The above problem, called the m-points inclusion problem is reducible to the intersection reporting problem in linear time by connecting m points together to form a non-convex non-simple polygon Q. A solution to the intersection reporting problem can easily determine whether a point is in a n-gon P, thereby solving the inclusion problem. Thus the intersection reporting problem must be at least $O(m \log n)$. □

2.2 The Intersection Detection Problem

The intersection detection problems for two simple non-convex n-gons and two convex n-gons have been studied, and have been solved in $O(n \log n)$ [Shamos and Hoey 1976] and $O(\log n)$ [Chazelle and Dobkin 1980] time, respectively. The algorithms contained in [Shamos and Hoey 1976] or in Section 2.1 solve the intersection detection problem for a convex n-gon and a non-convex m-gon in $O(m \log n)$ time. However, the time complexity can be further improved since both these

methods include some redundancy. In order to design a more efficient algorithm for this problem, it is necessary to preprocess $B(Q)$, so that both $B(P)$ and $B(Q)$ can be scanned sequentially, thus solving this problem in linear time.

Definition 2: Given a polygon Q and a point o, a particular point q on $B(Q)$ is *visible* from o, if the line segment (o,q) does not intersect another point of Q. A *visible chain* $V(Q,o)$ is a sequence of vertices (or edges) of Q which are all visible from o.

The following lemma shows that each edge of Q need not be compared against every edge of P in order to determine whether or not the two polygons intersect.

Lemma 1: *Given two polygons P, Q, a point o, $o \in R(P)$ and a visible chain $V(Q,o)$, then $R(P) \cap R(Q) \neq \emptyset$ iff $R(P) \cap V(Q,o) \neq \emptyset$ or $o \in R(Q)$.*

Proof: a) 'if part' is obvious.

b) 'only if part'. Assume $R(P)$ intersects $R(Q)$, and $o \notin R(Q)$. Then $B(Q) \cap R(P) \neq \emptyset$. For a convex polygon, there must exist at least one point $x \in B(Q) \cap R(P)$ such that x is visible from o. Therefore, $x \in V(Q,o)$ implies $R(P) \cap V(Q,o) \neq o$ /. □

By the above lemma, checking whether $V(Q,o) \cap R(P) = \emptyset$ is sufficient to answer the intersection detection problem. The

following procedure describes how to construct $V(Q,o)$, with two deques‘ , S and T. (Fig. 1)

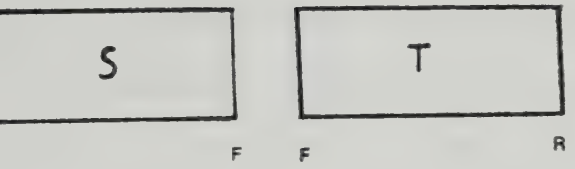


Fig 1

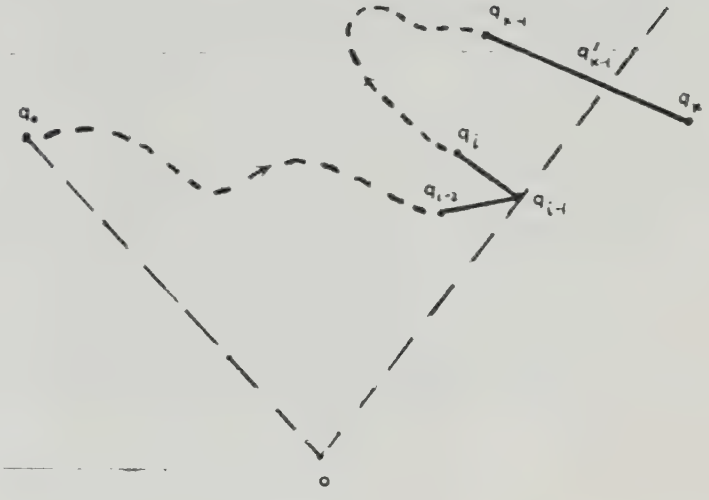


Fig 2

Vertices are stored in two deques S and T, where the concatenation of these two deques represents the portion of the visible chain already formed. Initially S and T are empty, and the vertices, $q_0,q_1,\ldots,q_{m-1}$ are pushed into S (or T) at F, as long as the polar angles of these vertices with respect to o are monotonically decreasing (or increasing). Assume q_i is the first vertex that backs up, (i.e., $\theta(q_i)>(<)\theta(q_{i-1})$). There are two cases to be considered.

Case 1: The edge (q_{i-1},q_i) is totally or partially invisible from o (Fig. 2). The following vertices $q_{i-1},q_i,\ldots,q_{k-1}$ are then removed until vertex q_k, where $k\geq i$, emerges from either end of the edge chain $(q_0,q_1,\ldots,q_{i-1})$ and becomes visible from o. If q_k emerges from the end associated

---

‘A deque is a list of elements where insertion and deletion can be done at both ends.

with q_{i-1}, as shown in Fig. 2, a new vertex q'_{K-1}, the intersection of the ray from o to q_{i-1} with (q_{K-1}, q_K), is pushed into S (or T) at F with q_K. If q_K emerges from the end associated with q_0, the vertices q'_{K-1} and q_K are pushed into S (or T) at R.

Case 2: The edge (q_{i-1}, q_i) is totally visible from o. Vertices $q_{i-1}, q_i, \ldots, q_{K-1}$, with $k>i$, are pushed into T (or S) at F as long as (a) their polar angles are monotonically increasing (decreasing) (Fig. 3) and (b) they are visible from o (Fig. 4).

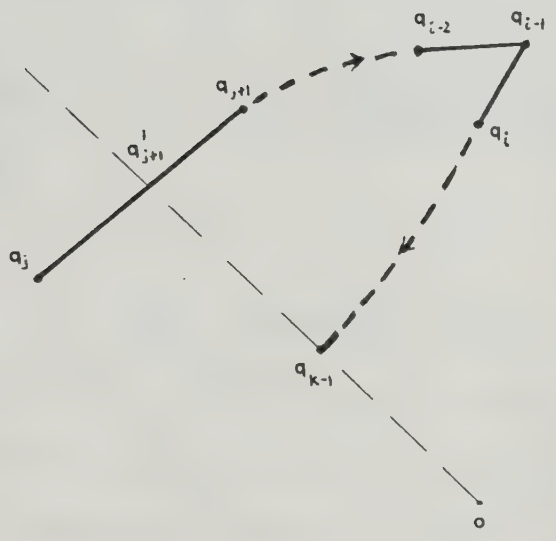


Fig 3

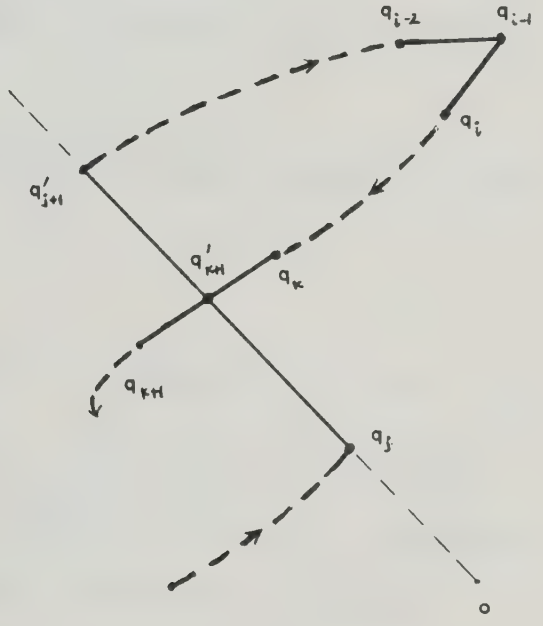


Fig 4

In any case, vertices $q_{i-1}, q_{i-2}, \ldots, q_{j+1}$ are popped from S (or T) at F until vertex q_j with $j<i-1$ is visible from o. These two subcases are considered separately.

a. With reference to Fig. 3, a new vertex q'_{j+1}, the intersection of the ray from o to q_{K-1} with edge (q_j, q_{j+1}), is pushed into S (or T) at F.

b. With reference to Fig. 4, a new vertex q'_{k+1}, the intersection of (q_k, q_{k+1}) with (q_j, q'_{j+1}), is pushed into T (or S) at F.

Even though special cases are considered in the above discussion, they have illustrated the key idea behind the algorithm. The above two cases are applicable to a similar environment in subsequent steps of the algorithm and are applied repeatedly with the possibility of the roles of S and T being interchanged. The algorithm terminates when all vertices in Q have been examined. It is easy to see that no vertex or edge is examined more than twice and that at most m new vertices are created. Thus the following lemma holds.

Lemma 2: *The visible chain $V(Q,o)$ can be constructed in $O(m)$* time.

In order to check whether $V(Q,o)$ intersects P, draw rays for P from $o \in R(P)$, partitioning the plane into n sectors; determine the sector to which each vertex $q \in V(Q,o)$, belongs; and determine whether $q \in R(P)$ or $V(Q,o) \cap R(P) = \emptyset$. By the definition of a visible chain and the convexity of a polygon, $V(Q,o)$ and P are monotonic with respect to the mapping θ. Thus vertices in $V(Q,o)$ and P can be scanned sequentially, without backtracking, to check whether $V(Q,o) \cap R(P) = \emptyset$. Note that the order of the vertices in $V(Q,o)$ must be reversed so that the vertices of $V(Q,o)$ and P are in a clockwise order with respect to o. This algorithm, which is similar to the

algorithm for merging two ordered sets, can be described bellow.

Algorithm 2: *Intersection detection problem for a convex n-gon and a non-convex m-gon.*

Input: *A non-convex polygon* $Q=(q_0,q_1,\ldots,q_{m-1})$ *and a convex polygon* $P=(p_0,p_1,\ldots,p_{n-1})$.

Output: *Yes, if they intersect. No, otherwise.*

Method:

```
1       choose a point o∈R(P) as the origin ;

2       if o∈R(Q)
          then return 'yes'
          else do
3               find and reverse  the order of
                V(Q,o)=( v0,v1,...,vk-1 ) ;
4               find j s.t. θ(p(j-1)mod n )≤θ(v0)≤θ(pj mod n ) ;
            end
        comment the index value of p is modulo n.

5       i←0 ;
6       while i≤k-1
           do
7           if θ(vi )>θ(pj )
              then do
                if pj is on the lefthand side of (vi-1 ,vi )
                    then return 'yes'
                    else j←(j+1)mod n;
                   end

8             else do
                if vi is on the righthand side of (pj-1 ,pj )
                    then return 'yes'
                    else i←i+1;
                   end

9          end
10      return 'no' ;
```

Theorem 2: *The intersection detection problem for a convex n-gon P and a non-convex m-gon Q can be solved in O(n+m) time.*

Proof: Steps 2 and 3 can be done $O(m)$ time. Step 4 takes $O(\log n)$ time by a binary search on $\theta(p_j)$. Since $V(Q,o)$ and P are monotonic with respect to the polar angle θ, steps 6 to 9 will be repeated no more than $m+n$ times. Thus, the total time is $O(m+n)$. □

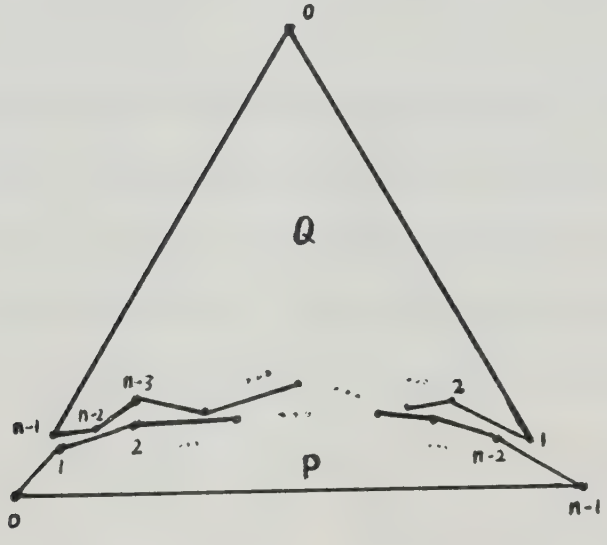


Fig 5

Theorem 3: *The intersection detection problem of a convex n-gon P and a non-convex n-gon Q requires at least O(n) operations in the worst case.*

Proof: Consider the case of two n-gons $P=(p_0,p_1,\ldots,p_{n-1})$, $Q=(q_0,q_1,\ldots,q_{n-1})$, with their vertices perturbed slightly from a unit circle and laid one after another alternatively as shown in Fig. 5. To be precise, $\theta(p_{i-1})\leq \theta(q_{n-i})\leq\theta(p_i)$ for $0<i<n$, and $r(p_i)=1+\epsilon_i$, $r(q_j)=1+\epsilon_j$, where

$|\epsilon_i|, |\epsilon_j| << 1$, and $i=0,1,\ldots,n-1$, $j=1,2,\ldots,n-1$. The lower bound can be derived from the number of comparisons necessary to decide whether a vertex is on the righthand or lefthand side of an edge. Obviously, to answer the intersection detection problem, every vertex in P and Q must be also checked, and its corresponding edge in polygon Q or P, (i.e., vertex p_{n-i} against edge (q_{i-1}, q_i) and vertex q_{n-i} against edge (p_{i-1}, p_i)). Therefore, the algorithm requires no less than $(2n-3)$ comparisons. □

2.3 The Intersection Detection Problem for two Nested Polygons

An optimal algorithm for the intersection detection problem of convex and non-convex polygons has been presented. However, if $R(P) \cap R(Q) \neq \emptyset$, this algorithm will not indicate whether or not one polygon is totally contained in the other. In order to solve this problem, the previously presented algorithm must be modified. Instead of constructing the visible chain $V(Q,o)$, find the *farthest chain* $F(Q,o)$ from a point o, where $F(Q,o)$ is defined as the set of points, $F(Q,o) = \{x \mid x \in Q, x$ is the farthest point from o at a certain orientation$\}$. A chain can be represented as a set of k vertices. Farthest chains can be constructed in linear time in the same way that visible chains are constructed. In order to check whether $R(P) \subset R(Q)$ or $R(Q) \subset R(P)$, it is necessary to determine whether $V(Q,o)$ and $F(Q,o)$ are both inside or outside of P. Obviously, this can be done in optimal linear time.

Theorem 4: *Given a convex n-gon and a non-convex m-gon, detecting whether* $R(P) \subset R(Q)$ *or* $R(Q) \subset R(P)$ *takes* $O(n)$ *time.*

Chapter 3

Minimum Distance Problem

If there is no intersection between a convex n-gon P and a non-convex m-gon Q, or between two convex (respectively, two non-convex simple) n-gons, then finding the minimum distance between them may be interesting. By modifying the algorithm given in Section 3.2, the minimum distance between P and Q can be determined in $O(m\log n)$ time. The minimum distance between a convex n-gon and a non-convex m-gon can be solved in $O(n+m)$ time by a monotonic property as shown in Section 2.2. (details are given in Section 3.1.1). In Section 3.1.2, an $O(\log n)$ algorithm finding the minimum distance of two separable convex n-gons is presented. An $O(n\log n)$ algorithm for finding the minimum distance of two separable simple non-convex polygons is also presented.

If the endpoints of minimum distance must be the vertices of given polygons, then the problem becomes more complicated. In Section 3.2.1, an optimal algorithm finding the minimum vertex distance between two separable convex n-gons is presented. In Section 3.2.2, the minimum vertex distance between non-intersecting a convex n-gon and a non-convex m-gon is considered. Finally, in Section 3.3, these problems are extended to the L_1 metric.

3.1 The Minimum Distance between the Boundaries of two Polygons

Definition 3: Let $d(x,y)$ be the Euclidean distance between two points x and y. Let P and Q be two polygons. The minimum distance between P and Q is defined as $d(P,Q)=\text{Min}\{d(x,y) \mid x\epsilon B(P),\ y\epsilon B(Q)\}$. The minimum distance between point x and Q is defined as $d(x,Q)=\text{Min}\{d(x,y) \mid y\epsilon B(Q)\}$. If there exist two points p, q, such that $p\epsilon P$, $q\epsilon Q$, and $d(p,q)=d(P,Q)$, then p, q are called a *pair of closest points* for P and Q.

Two polygons are called *separable*, if there exists a straight line l with the property that every vertex of P_1 lies on one side of l and every point of P_2 lies on the other.

3.1.1 Minimum Distance between a Convex n-gon and a Non-convex m-gon

It will be shown that the minimum distance between P and Q is the same as the minimum distance between $V(Q,o)$ and P, where $o\epsilon R(P)$.

Lemma 3: *Let p and q be a pair of closest points for P and Q. If P is convex then q is always visible from any point in $R(P)$.*

Proof: By contradiction. Assume q is not visible from a point $k\epsilon R(P)$. There must exist a point $q'\epsilon Q$ between k

and q. As shown in Fig. 6, a line parallel to the line segment (p,q) and intersecting with the line segment is drawn such that it passes through q' and intersects with the line segment (p,k) at p'. Obviously, $d(q,p)>d(q',p')$. Since $p,k\epsilon R(P)$, by the convexity of P, $p'\epsilon R(P)$, $d(q',p')\geq d(q',P)$, contradicting $d(P,Q)=d(p,q)$. □

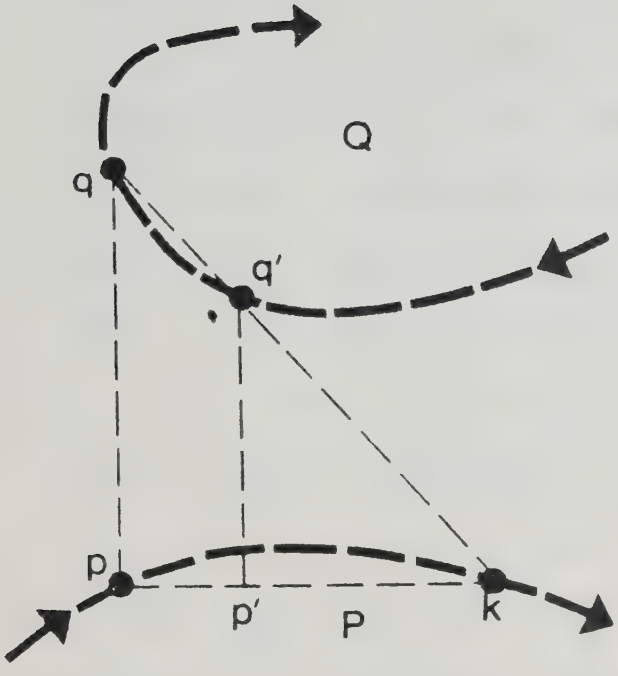


Fig 6

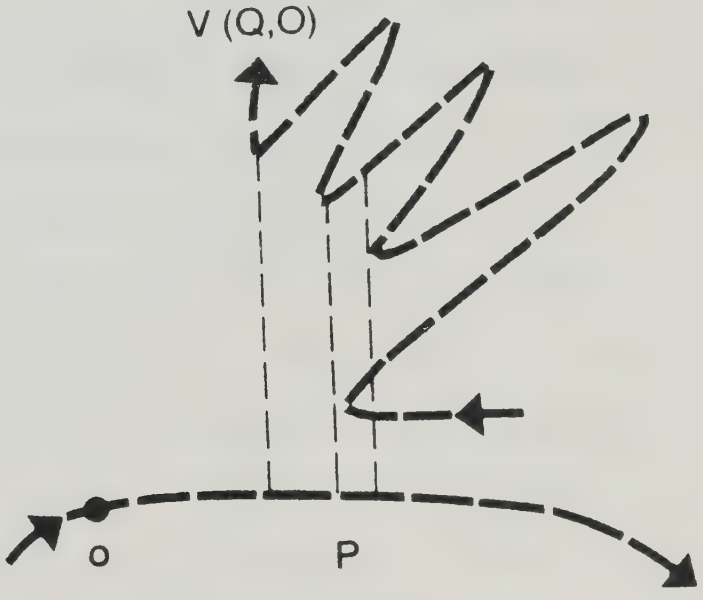


Fig 7

It is easily shown that there always exists a pair of closest points between P and Q such that one of them is a vertex of P or Q. Given Lemma 3, one can conclude that $d(P,Q)=d(P,V(Q,o))=$ Min$(\{d(p,V(Q,o))\mid p\epsilon P\}\cup\{d(q,P)\mid q\epsilon V(Q,o)\})$. There exist situations, indicated in Fig. 7, for

which finding the minimum distance between $V(Q,o)$ and P may take $O(m \log n)$ time. This can occur because if one scans $V(Q,o)$ sequentially, as in Section 2.2, to find $d(q,P)$ for some q may take $O(\log n)$ time using a binary search on the sectors of P. In order to achieve the linear time bound, one must establish some kind of monotonic property for the visible chain V of Q from P. Let $x,y \epsilon V$, x and y be ordered in such a way that $x>y$ if x is traced before y along V. Thus if $V=(v_0,v_2,\ldots,v_{k-1})$, then $v_0>v_1>\ldots>v_{k-1}$. Vertices of P are ordered similarly. One wants to construct a visible chain V of Q, which is monotonically decreasing with respect to the mapping ω_p, where ω_p maps a point $x \epsilon V$ to its closest point in $B(P)$, (i.e., if $x,y \epsilon V$ and $x>y$, then $\omega_p(x) \leq \omega_p(y)$, where $\omega_p(x),\omega_p(y) \epsilon B(P)$.) Thus $\omega_p(v_0) \leq \ldots \leq \omega_p(v_{k-1})$. Moroever, if there is a vertex $p \epsilon B(P)$ such that $\omega_p(v_i) \leq p \leq \omega_p(v_{i-1})$, then $v_i \geq \omega_p(p) \geq v_{i+1}$. Hence, if one can construct a visible chain V such that it has the above monotonic property with respect to the mapping ω_p, V can be scanned sequentially and $d(V,P)$ can be found in linear time. Note that the direction of V must be reversed.

The *extreme vertices* of P, are those vertices in P, which intersect a bounding rectangle of any fixed orientation such that no more than one vertex of P is contained in each edge of the rectangle. Without loss the generality, if the vertices of P are represented as Cartesian coordinates, the set of vertices in P with the maximum or minimum x or y coordinates are taken as the extreme vertices of P. The following lemma

shows that $V(Q,S) = \bigcap_{x \in S} V(Q,x)$, the set of points in Q which are visible from all points in S, is monotonically decreasing with respect to mapping ω_p, where S is the set of extreme vertices of P.

Lemma 4: *$V(Q,S)$ is monotonically decreasing with respect to ω_p, if S is a set of extreme vertices of P.*

Proof: In order to prove that $V(Q,S)$ is monotonically decreasing with respect to ω_p, one must show that for any two points $x,y \in V(Q,S)$, if $x>y$ then $\omega_p(x) \leq \omega_p(y)$. It is sufficient to prove that (a) edges $(x,\omega_p(x))$ and $(y,\omega_p(y))$ do not intersect, and (b) for any $x \in V(Q,S)$, $(x,\omega_p(x))$ never intersects with $V(Q,S)$.

a. Assume $(x,\omega_p(x))$ and $(y,\omega_p(y))$ intersect (Fig. 8), then either $d(x,\omega_p(y)) < d(x,\omega_p(x))$ or $d(y,\omega_p(x)) < d(y,\omega_p(y))$, contradicting that $(x,\omega_p(x))$ and $(y,\omega_p(y))$ are pairs of closest points.

b. Without loss of generality, assume c_1 and c_2 are two adjacent extreme points for P (Fig. 9). Since x is visible from c_1 and c_2, $V(Q,S)$ can never lie inside triangle $\Delta_1 c_1 c_2 x$. It is easily shown that $(x,\omega_p(x))$ always lies inside triangle $\Delta_1 c_1 c_2 x$ and therefore never intersects $V(Q,S)$. □

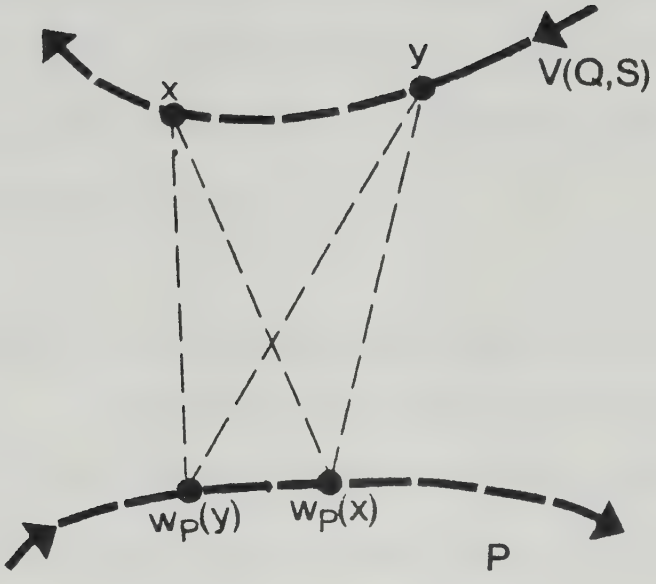


Fig 8

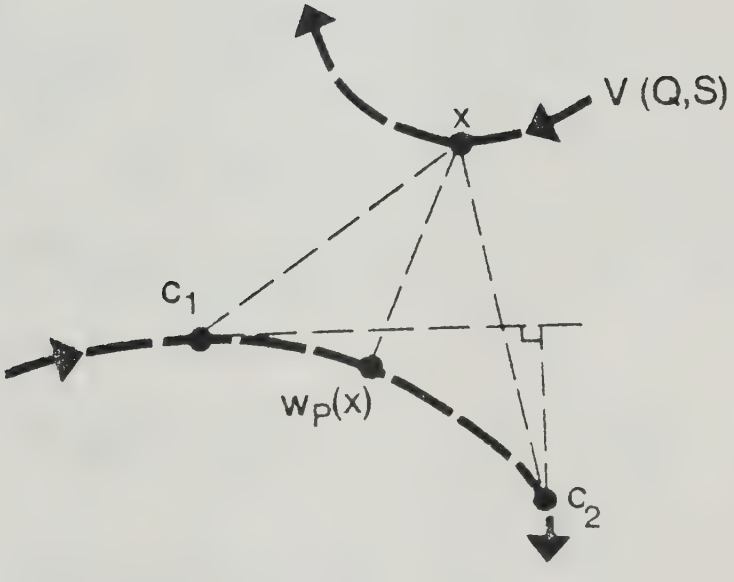


Fig 9

Similarly, P has the same monotonic property with respect to mapping ω_v. It is conjectured that S consists of no more than 3 points. Unfortunately it is not known whether there exists an easy method to find them. The algorithm is described as follows.

Algorithm 3: *Finding the minimum distance between a convex n-gon and a non-convex m-gon.*

Input: A convex n-gon P and a non-convex m-gon Q.

Output: The minimum distance and their closest points pair.

Method:

```
1   find four extreme vertices of convex n-gon P,
    and let S={c1,c2,c3,c4}

2   x←Q;
    for i←1 to 4 do x←V(x,ci) end;
    V(Q,S)←x;
      comment let V(Q,S)=(v0,v1,...,vk-1).

3   find j such that p(j-1) mod n ≤ ωp(v0) ≤ pj mod n ;

      comment the index value of p is modulo n.
    i←1; D←d(v0,(pj-1,pj));

4   while i≤k-1 do

5       if ωp(vi)≤pj
          then D←Min(d(vi,(pj-1,pj)),D);
               i←i+1;

6         else D←Min(d(pj,(vi-1,vi)),D);
             j←(j+1)mod n;

    end.
```

Theorem 5: *Algorithm 3 gives an optimal solution for finding the minimum distance between a convex n-gon and a non-convex m-gon in O(m+n) time.*

Proof: If consider the case shown in Theorem 1, and assume the basic operations are computing the minimum distance between a point and a line segment or between two points, then $O(m+n)$ operations are necessary. □

3.1.2 Minimum Distance between two Separable Convex n-gons

Consider the problem of finding the minimum distance between two separable convex n-gons P_1 and P_2 with n_1, n_2 vertices, respectively. The best known algorithm [Schwartz 1981] takes $O(\log^2 n)$ by performing double binary search on vertices of P_1 and P_2. Introduced here is an algorithm performing a single binary search on the chains of P_1 and P_2. Lemma 3 shows that if p_1 and p_2 are the pair of closest points of P_1 and P_2, respectively, then p_1 is visible from any point in $R(P_2)$ and similarly p_2 is visible from any point of $R(P_1)$. Therefore, the problem of finding the minimum distance between P_1 and P_2 can be reduced to the problem of finding the minimum distance between two visible chains V_1 and V_2 of P_1 and P_2, respectively. In order to design an efficient algorithm to find this minimum distance between P_1 and P_2 in $O(\log n_1 + \log n_2)$ time, the following problems must be solved:

1. V_1 and V_2 must be found in $O(\log n_1 + \log n_2)$ time (Lemma 5);
2. it must be shown that V_1 and V_2 have the unimodal property with respect to the mapping Δ_2 and Δ_1, respectively, given that $\Delta(x)$ is the minimum distance between V and x (Lemma 6);

3. the decreasing or increasing directions of the minimum distance between V_1 and V_2 must be found in constant time (Lemma 7);

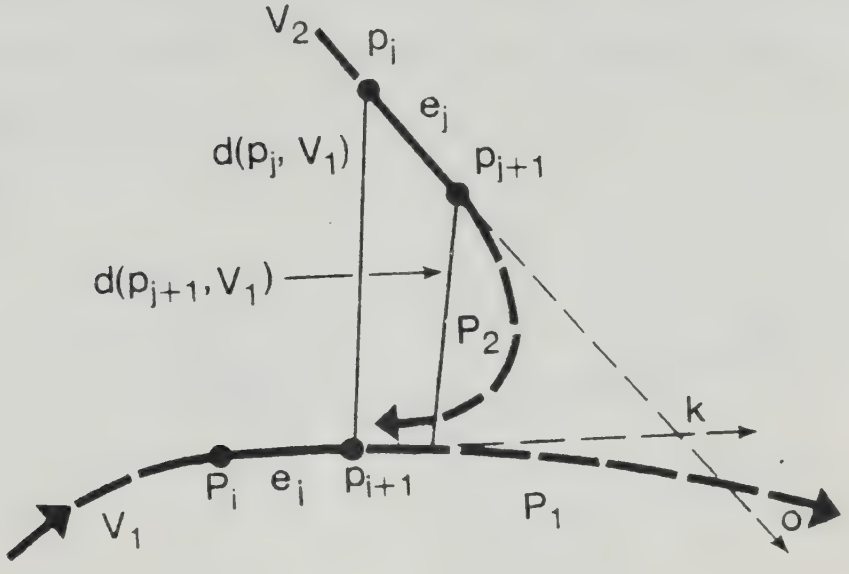


Fig 10

In order to solve the third problem, one must determine in constant time for any given edge (p_j, p_{j+1}) in V_2 whether or not $d(p_j, V_1) > d(p_{j+1}, V_1)$. The two extended edges, $e_i \in V_1$ and $e_j \in V_2$ are used to determine the increasing or decreasing direction of V_1 or V_2. If V_1 and V_2 are visible chains from a single origin, there are situations, as shown in Fig.10, for which the decreasing (increasing) direction of V_1 (V_2) cannot be decided in constant time. However, if $V_1 = V(P_1, S_2)$ and $V_2 = V(P_2, S_1)$, where S_1 and S_2 are sets of extreme points for P_1 and P_2, then it can be shown by Lemma 7, that the third problem is solvable in constant time.

Lemma 5: V_1 and V_2 can be found in $O(\log n)$ time.

Proof: Consider finding $V(P_1,o)$ for $o \epsilon P_2$. Since $B(P_1)$ is unimodal with respect to θ, the vertex with the maximum polar angle and the vertex with the minimum polar angle can be found in $O(\log n)$ time. Thus $V(P_1,o)$ and $V_1=V(P_1,S)$ can be found in $O(\log n)$ time. Similarly, this holds for V_2. □

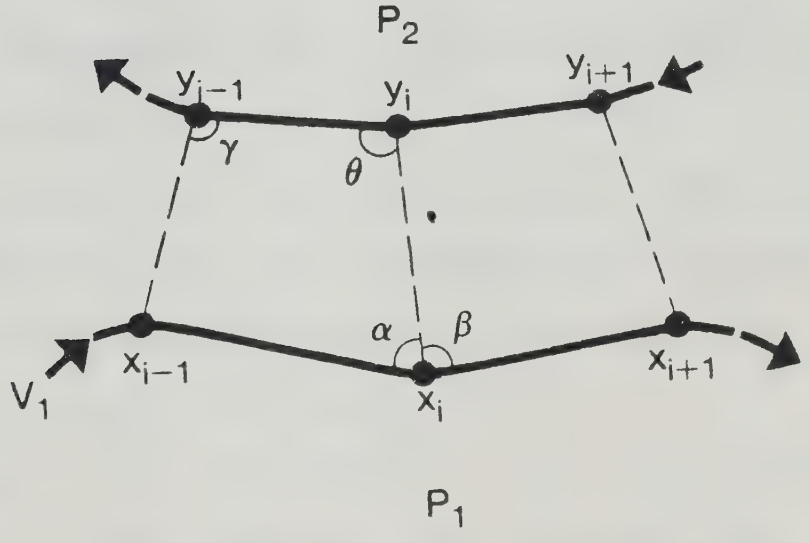


Fig 11

Lemma 6: V_1 (V_2) *is unimodal with respect to* Δ_2 (Δ_1), *where* $\Delta_1(x)=d(x,V_1)$, $\Delta_2(x)=d(x,V_2)$.

Proof: This lemma is proven by showing that for $x_{i+1}>x_i>x_{i-1}$,

where $x_{i-1}, x_i, x_{i+1} \epsilon V_1$, if $\Delta_2(x_{i-1}) < \Delta_2(x_i)$ then $\Delta_2(x_i) < \Delta_2(x_{i+1})$ and if $\Delta_2(x_{i+1}) < \Delta_2(x_i)$ then $\Delta_2(x_i) < \Delta_2(x_{i-1})$. Again these two cases are so similar that only the former case is considered. The proof is by contradiction. Assume that $\Delta_2(x_{i-1}) < \Delta_2(x_i)$ and $\Delta_2(x_i) \geq \Delta_2(x_{i+1})$. Let $y_i = \omega_{V_2}(x_i)$ be the closest point of V_2 from x_i. Now, from the definition of V_1 and Lemma 4, $y_{i+1} \leq y_i \leq y_{i-1}$ (Fig. 11), and because of the convexity of P_2, if $\Delta_2(x_{i-1}) < \Delta_2(x_i)$, then the angles $\gamma, \theta_1 \geq 90°$ and the angle $\alpha < 90°$. Moroever, $\Delta_2(x_i) \geq \Delta_2(x_{i+1})$ implies that angle $\beta \leq 90°$ and $\alpha + \beta < 180°$, contradicting the fact that P_1 is convex. □

Lemma 7: *Given a pair of edges, (p_i, p_{i+1}) and (q_j, q_{j+1}) in V_1 and V_2 respectively, one can determine the result of either of the following two comparisons in constant time, (1) $\Delta_2(p_i)$ vs $\Delta_2(p_{i+1})$ (2) $\Delta_1(q_j)$ vs $\Delta_1(q_{j+1})$.*

Proof: Let the two edges (p_i, p_{i+1}) and (q_j, q_{j+1}) be extended and let k be their intersection. Without loss of generality, assume that p_{i+1} is between p_i and k. Now, from the definition of V_1 and V_2, referring to Fig. 12, the region of P_2 can not be below the extended line (q_j, q_{j+1}), implying that (q_j, q_{j+1}) must point to the left. If a perpendicular is drawn from p_{i+1} to the extended line (q_j, q_{j+1}) intersecting at p', there are two cases to consider.

a. q_j is on the righthand side of p',(note that p' can be on the righthand or lefthand side of k (Fig. 12)). Then because of the convexity of P_2,

clearly $\Delta_2(p_i) > \Delta_2(p_{i+1})$.

b. q is on the lefthand side of p'. Let x be the closest point at V_1 from q_{j+1}. If $d(q_{j+1}, x) > d(q_j, x)$, then clearly $\Delta_1(q_{j+1}) = d(q_{j+1}, x) > d(q_j, x) \geq \Delta_1(q_j)$. On the other hand, if $d(q_{j+1}, x) \leq d(q_j, x)$, a line parallel to (q_{j+1}, x) can be drawn through q_j intersecting (x, p_{i+1}) at y. Then from the convexity of P_1, $y \in R(P_1)$ and $\Delta_1(q_{j+1}) = d(q_{j+1}, x) > d(q_j, y) \geq \Delta_1(q_j)$. This will hold even if the intersection angle $\alpha > 90°$. □

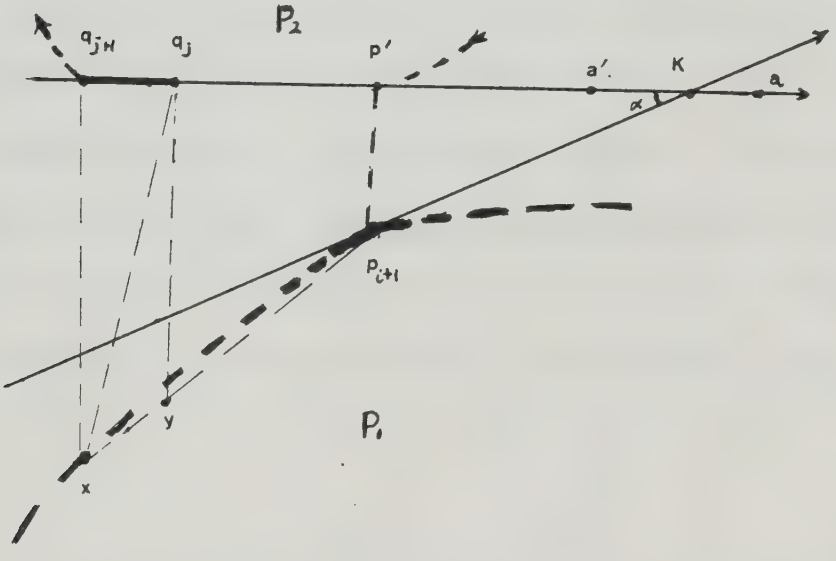


Fig 12

Given Lemma 7, one can reduce in constant time at least one of the chains V_1 or V_2 by half. This process can be repeated until only two edges, one for each V_i, remain. Thus, by a binary search, the minimum distance between P_1 and P_2 can be found in $O(\log n_1 + \log n_2)$ time. The algorithm (called

Algorithm 4) is omitted. The following theorem proves the optimality of this algorithm.

Theorem 6: *Given two separable convex n-gons P_1, P_2, there exists an asympotically optimal algorithm which can find the minimum distance between them in $O(\log n)$ time.*

Proof: It remains to be shown that the minimum distance problem requires at least log n comparisons. Consider a finite set of positive numbers $X=\{x_0,x_1,\ldots,x_{n-1}\}$. If the values of the numbers are unimodal, (i.e., there exists an integer j, $0\leq j<n$, such that $x_{i-1}>x_i$ for all $0<i\leq j$ and $x_{i-1}<x_i$ for all $j<i\leq n$) then a lower bound can be found using reducing this to a problem, which requires at least log n comparisons by information theory. Furthermore, one can easily show that this problem is reducible to the minimum distance problem in constant time. Thus the minimum distance problem is at least $O(\log n)$ time. □

Corollary: *the line separating two convex n-gons can be found in $O(\log n)$ time.*

3.1.3 Minimum Distance between two Nested Polygons

Two optimal algorithms to find the minimum distance between two convex n-gons or between a convex n-gon and a non-convex m-gon have be presented. In the case of one polygon entirely containing the second, if one disregards the

fact that these two polygons do intersect each other, then the time complexity of finding the minimum distance between their boundaries is linear with respect to the number of vertices in the polygons. The time complexity of finding the minimum distance between two convex n-gons will increase from $O(\log n_1+\log n_2)$ to $O(n_1+n_2)$ since the minimum distances between two convex n-gons are multimodel with respect to the mapping, which maps each vertex of one polygon to its closest point in the other polygon. Since they are monotonic with respect to the polar angle with an origin inside the polygon, the $O(n_1+n_2)$ algorithm, finding the minimum distance by sequentially checking the two boundaries without backtracking, is not difficult to design. For finding the minimum distance between a convex n-gon P and a non-convex m-gon Q, where $R(Q)\subset R(P)$, the time complexity will remain unchanged. This is easy to show by constructing a convex hull for Q, and using the same method for the case of one convex polygon including the second convex polygon. The endpoint of the minimum distance in Q must be a vertex of its convex hull.

Theorem 7: *Given two convex n-gons P_1, P_2 such that $R(P_2)\subset R(P_1)$ or $(R(P_1)\subset R(P_2))$ the minimum distance between them can be found in $O(n_1+n_2)$ time.*

Theorem 8: *Given a convex n-gon P and a non-convex m-gon Q such that $R(Q)\subset R(P)$, the minimum distance between them can be found in $O(n+m)$ time.*

3.2 The Minimum Vertex Distance between two Polygons

In this section, the studies are restricted to the minimum vertex distance between two separable planar convex polygons.

Definition 4: Let $P_1 \epsilon (p_0, p_1, \ldots, p_{n-1})$ and $P_2 \epsilon (q_0, q_1, \ldots, q_{n-1})$, and let $d(P_1, P_2)$ be the *minimum vertex distance* between the two vertex sets of P_1 and P_2, i.e $d(P_1, P_2) = \min\{d(p,q) \mid p \epsilon P_1$ and $q \epsilon P_2\}$. Two vertices $p \epsilon P_1$ and $q \epsilon P_2$ are called *closest vertices* of P_1 and P_2 if $d(p,q) = d(P_1, P_2)$.

The *minimum vertex distance problem* is to find the minimum vertex distance and its closest vertices between two sets of vertices. If P_1 and P_2 are considered as two arbitrary sets of vertices, rather than as sequences of vertices from two convex polygons, then by creating the *Voronoi* diagrams, one can find the minimum vertex distance between P_1 and P_2 in $O(n \log n)$ time. In this section, an $O(n)$ time optimal algorithm to solve the minimum vertex distance problem for two separable convex n-gons is presented.

3.2.1 Minimum Vertex Distance between two Separable Convex n-gons

Definition 5: A *vertex chain* $\langle x,y \rangle$ of P_1 is defined as an ordered set of points $(x, p_i, p_{i+1}, \ldots, p_j, y)$ where $(p_i, p_{i+1}, \ldots, p_j)$ is an ordered subsequence of $P_1 = (p_0, \ldots, p_{n-1})$, and x and y are points on the edges (p_{i-1}, p_i) and (p_j, p_{j+1}), respectively. An *edge chain* (x,y) of P_1 is set of edges

represented by the vertex chain $\langle x,y\rangle$, i.e., $\{(x,p_i),(p_i,p_{i+1}),\ldots,(P_j,y)\}$. A *visible edge chain* (x,y) of P_1 from q is defined as the part of the boundary of P_1 that is visible from q, i.e., $\{s|s\epsilon B(P_1)$ and line segment (q,s) does not contain any other points in $B(P_1)\}$.

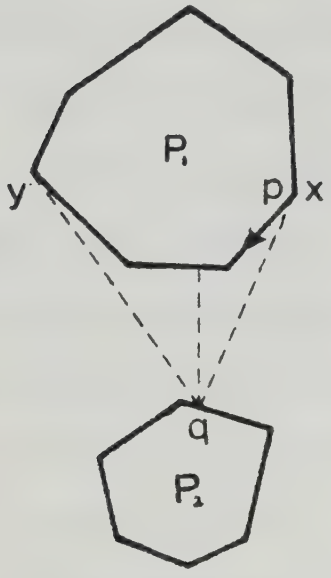


Fig 13

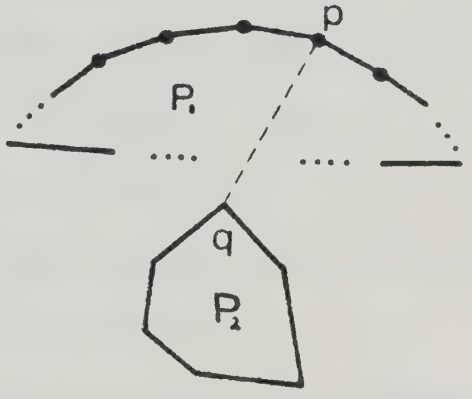


Fig 14

Consider a convex polygon P_1 and a point $q\epsilon B(P_2)$. In Fig. 13, one can see that the minimum distance (not the vertex distance) between q and P_1 is the same as the minimum distance between q and the visible edge chain (x,y) of P_1 from q. Moreover, as p traces along this edge chain (x,y), $d(q,p)$ will decrease gradually until some point in (x,y) is reached, from there on $d(q,p)$ begins to increase (i.e., $d(q,p)$ is unimodal with respect to $p\epsilon(x,y)$). Because of this unimodal property and the monotonic property between two visible chains, the minimum distance between two convex polygons P_1 and P_2 can be found in less than $O(n)$ time. However, this unimodal property

is no longer valid if the *vertex distance* between q and the convex polygon P_1, $d(q,P_1)$, is to be considered. For example, in Fig. 14, $d(q,p)$ is *multimodal* if p is taken at each of the *vertices* of P_1 in sequence. Because of this, no algorithm can solve this minimum vertex distance problem between two convex polygons in less than $O(n)$ time. In the following, an $O(n)$ algorithm for solving this minimum vertex distance problem for two separable convex polygons is presented.

First, partition the vertex set of polygon P_1 into vertex chains, P_{11} and P_{12}, and polygon P_2 into P_{21} and P_{22}. Now if the minimum vertex distances $d(P_{11},P_2)$, $d(P_{21},P_1)$ and $d(P_{12},P_{22})$ can be found in $O(n)$ time, then the minimum vertex distance problem can be solved in $O(n)$ time.

The polygon P_1 is partitioned as follows: A vertex $q_\alpha \epsilon P_2$ is chosen such that if $p' \epsilon B(P_1)$ has the property that p' is the closest point of $B(P_1)$ from q, i.e. $d(q_\alpha,p')=d(q_\alpha,P_1)$ (note that p' may not be a vertex of P_1), then $q_{\alpha-1}, q_{\alpha+1}$ are on one side of the line (q_α,p'). In fact all the vertices of P_2 are on the same side of (q_α,p'). Then there is at least one other pair of points say $q_\beta \epsilon P_2$ and $p'' \epsilon B(P_1)$, having the same property that $d(q_\beta,p'')=d(q_\beta,P_1)$, where all the vertices of P_2 are on the other side of (q_β,p'') (Fig. 15).

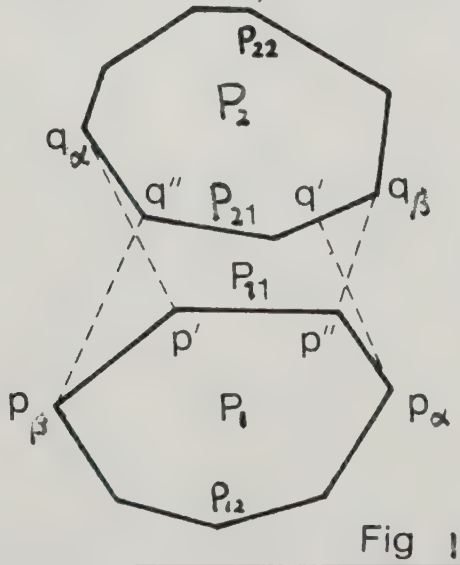


Fig 15

The polygon P_1 is partitioned by these two points p' and p''. The set of vertices lying between p' and p'' which face P_2 are called $P_{11}=(p',p'')$ and the remaining set of vertices of P_1 are called $P_{12}=(p'',p')$. Note that p' and p'' may not be in P_{11} or P_{12} if p' and p'' are not vertices of P_1. Polygon P_2 is partitioned by q' and q'' similarly, i.e. $P_{21}=(q',q'')$ and $P_{22}=(q'',q')$ as shown in Fig. 15.

The line segments representing the shortest distances from a set of vertices to P_1 never intersect, by lemma 4. After the shortest distance line segment from an arbitrary vertex in P_2 *to* $B(P_1)$ is found in $O(\log n)$ time, the remaining shortest distance line segments from vertices of P_2 to $B(P_1)$ can be found in $O(n)$ by examining the vertices in P_2 sequentially and scanning the boundary of P_1 accordingly. Since testing whether or not all the vertices of P_2 are on one side of any shortest distance line can be done in constant time, p' and p'' can be found in no more than $O(n)$ time. The same is true for the points q' and q'' in $B(P_2)$.

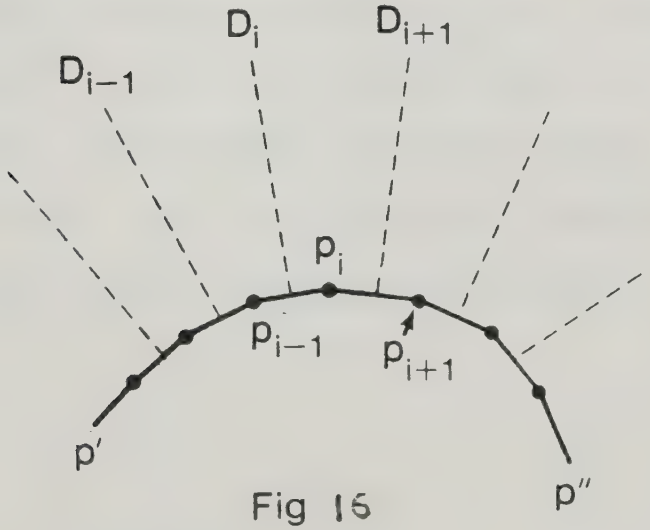


Fig 15

Lemma 8: *The minimum vertex distance problem between $P_{1,1}$ and P_2 can be found in $O(n)$ time.*

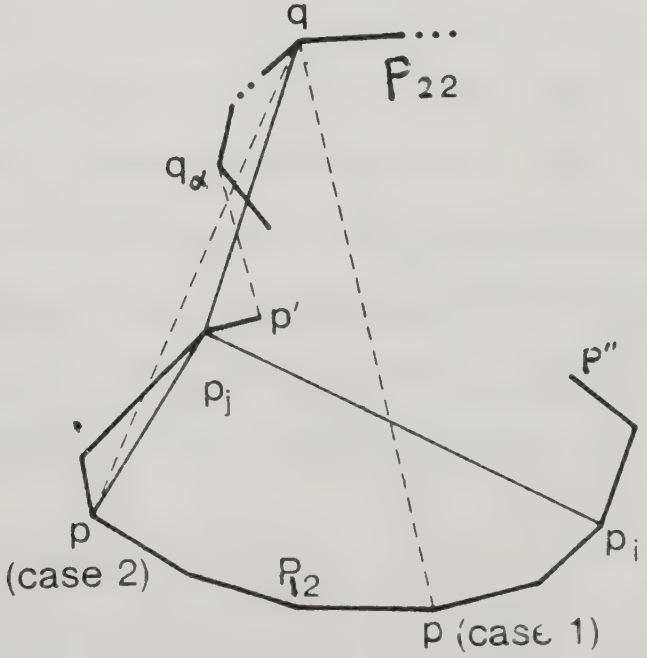
Proof: For each edge in the edge chain represented by $P_{1,1}$, say (p_{i-1}, p_i), a perpendicular bisector D_i can be constructed in constant time (Fig. 16). These perpendicular bisectors for the edges in $P_{1,1}$ partition the plane into a number of regions. Because of the convexity of $P_{1,1}$, it is easily shown that for any point x lying within the region bounded by $D_{i+1}, D_i, (p_{i-1}, p_i)$ and (p_i, p_{i+1}), $d(x,P_1)$ must be equal to $d(x,p_i)$. Note that some of these points, say y may not be bounded within two perpendicular bisectors, in which case $d(y, p_1)$ must be equal to the distance between y and one of the end points of p_1. For the first vertex q_0 of P_2, finding the region to which q_0 belongs may take $O(\log n)$ time. Finding the regions which contain the other vertices is done by processing each vertex sequentially and scanning the regions in order. Since there are at most n vertices and two scannings of the regions (forward and backward), the minimum vertex distance and the closest vertices between $P_{1,1}$ and P_2 can be found in $O(n)$ time. □

Similarly, the minimum distance problem between $P_{2,1}$ and P_1 can be solved. The following two lemmas describe how the shortest vertex distance between $P_{1,2}$ and $P_{2,2}$ is found in $O(n)$ time.

Lemma 9: *Let p_i and p_j be two arbitrary vertices in P_{12} and $q \epsilon P_2$, where the angle $\langle qp_j p_i$ is greater than 90°, then the line segment (q,p_j) is the minimum vertex distance line between q and the vertex chain $\langle p_i, p_j \rangle$.*

Proof: Consider some vertex p in the vertex chain $\langle p_i, p_j \rangle$, as shown in Fig. 17. Because of the convexity of P_1, p must be on the other side of line (p_j, p_i) with respect to q. There are two cases to consider:

1. Line segment (q,p) intersects line segment (p_j, p_i); then $d(q,p) > d(q,p_j)$, since angle $\langle qp_j p_i$ is greater than 90°.
2. Line segment (q,p) does not intersect line segment (p_j, p_i); then since $q \epsilon P_{22}$, and $p \epsilon P_{12}$, q and p must be on the opposite sides of line (q_α, p'). This implies that the angle $\langle qp_j p$ must be greater than or equal to 90° because of the convexity of P_{12} and the properties of the line segment (q_α, p'). Thus $d(q,p) > d(q,p_j)$. Therefore, $d(q,p_j)$ is indeed the shortest vertex distance between q and vertex chain $\langle p_i, p_j \rangle$. □



Lemma 10: *The minimum vertex distance between P_{12} and P_{22} can be found in $O(n)$ time.*

Proof:

This lemma is proven by construction. Given $P_{12}=(p'',p_i,\ldots,p_j,p')$ and $P_{22}=(q'',q_i,\ldots,q_j,q')$. First consider the quadrangle $q_i q_j p_i p_j$ as shown in Fig. 18. One of the interior angles must be $\geq 90°$. Without loss of generality, assume that angle $\angle q_i p_j p_i$ is greater than or equal to 90°. From Lemma 9, it can be concluded that $d(q_i,p_j)$ is the shortest vertex distance from q_i to P_{12} Let this be the initial value for $d(P_{12},P_{22})$. Now remove vertex q_i from P_{22}, the new quadrangle $q_j q_{i+1} p_j$ and p_i, where q_{i+1} is the vertex in P_{22} adjacent to q_i, is then considered. Again one of these four interior angles must be larger than 90°. The minimum vertex distance from one of these vertices to its opposite vertex chain can now be determined. This new minimum distance is compared with the old $d(P_{12},P_2)$, $d(P_{12},P_2)$ and updated, after which one of these vertices is removed. This process is repeated where the shortest vertex distance $d(P_{12},P_2)$ is updated at each step until only two vertices remain. Since the removed vertices are those vertices whose shortest distances to their opposite vertex chains have already been considered, the final answer for $d(P_{12},P_{22})$ is the minimum distance between P_{12} and P_{22}. As far as the time complexity is concerned, the

interior angle which is $\geq 90°$ can be identified and the corresponding vertex can be removed in constant time. Since there are at most $O(n)$ vertices in P_{12} and P_{22}, the minimum vertex distance between P_{12} and P_{22} is found in $O(n)$ time. □

The algorithm (Algorithm 5) is easy to design. It is not given here. Thus the following theorem holds.

Theorem 9: *The minimum vertex distance problem between two convex separated polygons can be solved in $O(n)$ time.*

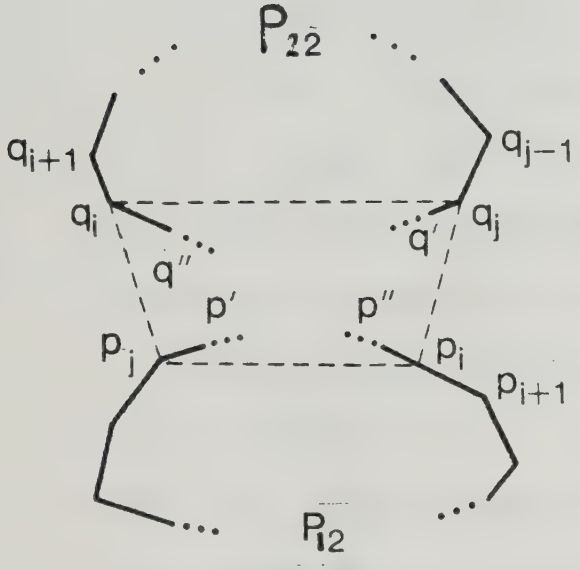


Fig 18

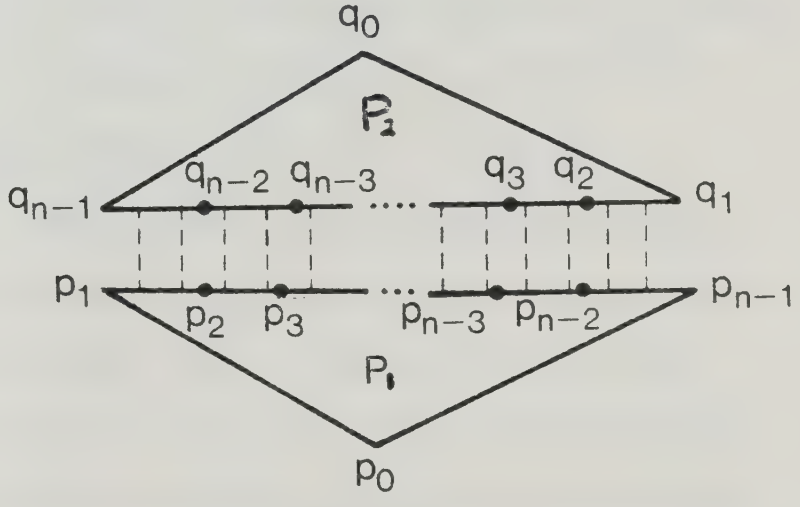


Fig 19

Theorem 10: *The minimum vertex distance problem for two separable convex n-gons requires at least (n-1) comparisons.*

Proof: Consider the case of two convex n-gons, $P_1=(p_0,\ldots,p_{n-1})$ and $P_2=(q_0,\ldots,q_{n-1})$ as shown in Fig. 19. The two

chains $(p_1,\ldots,p_{n-1})$ and $(q_1,\ldots,q_{n-1})$ are parallel to each other and also to the x-axis. (i.e. all the p_i's have the same y-coordinate while all the q_i's have another y-value) Part of the x-axis is divided into $2n-3$ equal segments. The x-coordinate of q_1 must lie within the first segment, q_i in the $(2i-1)$st segment and q_{n-1} in the $(2n-3)$rd segment. Similarly for the p_i's, the x-coordinate of the p_i must lie within the $((2n-1)-2i)$th segment. One can see that $d(q_i,P_1)=d(q_i,p_{n-i})$ and thus the minimum vertex distance between P_1 and P_2 must be $min\{d(q_i,p_{n-i}) \mid i=1,\ldots,n-1\}$. Clearly, in order to find the minimum vertex distance between P_1 and P_2, $(n-1)$ comparisons are required in the worst case. □

With the above lemma, the proposed $O(n)$ algorithm for solving the minimum vertex distance is asymptotically optimal.

The *nearest neighbor* problem for a convex polygon has been solved by Lee and Preparata [Lee and Preparata, 1977] in $O(n)$. By using Lemma 10, a simpler algorithm for solving the above problem with $O(n)$ time complexity can be designed.

3.2.2 Minimum Vertex Distance between a Convex n-gon and a Non-convex m-gon

Consider the following problem:

Given a convex n-gon P and a non-convex m-gon Q which do not intersect. Find the *minimum vertex distance* between them such that the closest vertices are visible.

In the above problem, the *closest vertex* in Q need not be visible from the *extreme points* in P; therefore, the problem is different from that in Section 3.1.1. The above problem is also different from that in section 3.2.1, where the closest vertices may not be visible from each other. In order to solve this problem based on the concept of visibility, the definition of a visible chain must be slightly modified. If an area is bounded by D_{i+1}, D_i and (p'_i, p_i), (p_i, p'_{i+1}), then it is called *region* i, where p_{i-1}, p_i, p_{i+1} are three consecutive vertices in P, and D_i, D_{i+1} are perpendicular bisectors (refer to Fig. 16). The minimum vertex distance from an arbitrary vertex $q_j \epsilon Q$ to convex n-gon P is $d(q_j, p_i)$ if q_j lies inside the *region* i (possiblly, the minimum vertex distance is $d(q_j, p_{i-1})$ (resp. $d(q_j, p_{i+1})$) if q_j lies on D_i (D_{i+1}), respectively.) If Q_i is the part of the polygon Q, which is in region i, $V(Q_i, p_i)$ is the visible chain of Q from p_i which is in region i. Define $V(Q,P) = \bigcup_{p_i \in P} V(Q_i, p_i)$, i.e., $V(Q,P)$ is the concatenation of all the chains $V(Q_i, p_i)$. It can be proven that this chain $V(Q,P)$ has the monotonic property with respect to mapping ω, mapping each vertex of $V(Q,P)$ to its closest vertex of P. It can be shown that if p_i and q_γ are two closest vertices, then both p_i and q_j must be in the same region, and q_j is visible from the vertex p_i. One can now conclude that the closest vertex in Q is included in $V(Q,P)$ since all visible points of Q from i are also included in $V(Q, p_i)$. A procedure constructing $V(Q,P)$ in $O(m \log n)$ time is similar to that in Section 3.2. Finding the region for which

an arbitrary vertex q_j lies is $O(\log n)$ if a binary search on the n regions of P is used.

Start at the vertex q_0 of Q, and find the corresponding regions of two consecutive vertices, say h and l . Connect q_0, q_1, draw a ray from l to q_1, and decide the visibility of line segment (q_1,q_2) from p_ℓ according to the angle of line (q_0,q_1), line (q_1,q_2) and ray (p_ℓ,q_1). The region on which the vertex q_s, an arbitrary vertex q_s of Q, lies is determined. Then the visibility of line segment (q_{s-1}, q_s) from p_r is found. Only two cases can occur, which are similar to these in Section 3.2. For each q_s of Q, it takes $O(\log n)$ time to find its corresponding region. Since no vertex or edge of Q is considered more than twice, only m vertices are created, thus the following lemma holds.

Lemma 11: *A visible chain $V(Q,P)$ can be constructed in $O(m\log n)$ time.*

The proof of monotonic property between $V(Q,P)$ and P is similar to Lemma 4 in Section 3.1.1. Let $d(q_i,P)$, $d(q_{i+1},P)$ be two consecutive minimum vertex distances, then the line segments (q_i,p_j) and (q_{i+1},p_s) do not intersect each other except that $p_j = p_s$. This property holds for the case of $d(p_i,V(Q,P))$, $d(p_{i+1},V(Q,P))$. The rest of the proof is omitted.

Therefore, one can design a procedure to check the minimum vertex distance between $V(Q,P)$ and P, in $O(n+m)$ time.

Theorem 11: *The minimum vertex distance between a convex n-gon and a non-convex m-gon can be found in time O(mlog n).*

The lower bound for finding the minimum vertex distance between a non-intersecting convex *n*-gon and a non-convex *m*-gon remains an unsolved problem.

3.2.3 Minimum Vertex Distance for two Nested Polygons

Consider two convex polygons as two vertex sets, i.e., the minimum vertex distance may cross the boundary of the polygons. In this case, if two convex polygons intersect each other, it is not known whether the minimum vertex distance can be found in $O(n)$ time. If a convex *n*-gon entirely contains another convex *n*-gon and the closest vertices are visible, then solving the minimum vertex distance problem can be based on following theorem.

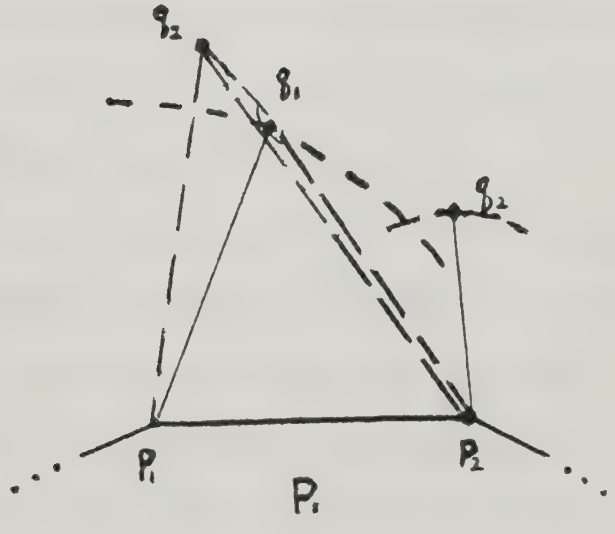


Fig. 20

Theorem 12: *Let P_1, P_2 be two vertex chains such that $R(P_1) \subset R(P_2)$. The minimum vertex distance between P_1 and P_2 can be found in $O(n)$ time.*

Proof: Assume $d(p_1,P_2)=d(p_1,q_1)$ and $d(p_2,P_2)=d(p_2,q_2)$. Therefore, $d(p_1,q_1)>d(p_1,q_2)$ and $d(p_2,q_2)<d(p_2,q_1)$. But $d(p_2,q_2)$ can not intersect $d(p_1,q_1)$ as shown in Fig. 20. P_1 is monotonic with respect to the mapping ω, which maps each vertex of P_1 into its closest vertex in P_2. Therefore, the minimum vertex distance between P_1 and P_2 can be found in O(n) time.□

3.3 The Minimum Distance between two Polygons in the L_1 Metric

In the previous sections, the minimum distance between two separable convex polygons, or between a non-intersecting convex polygon and a non-convex polygon was considered in the *Euclidean* (i.e., the L_2) metric. The minimum distance in metric s is defined as follows:

Let the two points p and q in the Euclidean plane with coordinates (Xp,Yp) and (Xq,Yq), respectively, and a real number s, where $1 \leq s < \infty$ be given.

The distance between p and q in the Ls metric is defined as $ds(p,q)=(|Xp-Xq|^s+|Yp-Yq|^s)^{1/s}$ and $d_\infty(p,q)=Max(|Xp-Xq|,|Yp-Yq|)$.

Here, the minimum distance problem of two separable convex polygons is discussed only in the L_1 metric since there does not appear to be any practical significance in the general s metric. The problems are presented as follows:

Given two separated convex n-gons P_1 and P_2 in the Euclidean plane, let c_1, c_2 be the two closest points with minimum distance $d(P_1,P_2)$. Also let $R(P_1)$, $R(P_2)$ denote the interior of polygon P_1 and P_2, respectively, and V_1 and V_2 the corresponding sets of vertices. The minimum distance of P_1 and P_2 in the L_1 metric is

$d(P_1,P_2)=min\{d(x,y) \mid x\epsilon P_1,\ y\epsilon P_2 \text{ and } d(x.y)\cap R(P_1)\cap R(P_2)=\emptyset\}$.

The minimum vertex distance of P_1 and P_2 in the L_1 metric is

$d(P_1,P_2)=min\{d(x,y) \mid x\epsilon V_1,\ y\epsilon V_2\}$.

It will be shown that the minimum distance of two convex polygons can be found in $O(\log n)$ time, and the minimum vertex distance in $O(n)$ time.

The definition of visibility will be slightly modified so that it is consistant with the L_1 metric.

Definition 6: A point $p\epsilon P_1$ is L_1 *visible* from $q\epsilon P_2$ if there exists a pair of connected line segments $((q,s),(s,p))$ such that (q,s) is parallel to the x axis and (s,p) is parallel to the y axis or vice versa, and $((q,s),(s,p))\cap R(P_1)=\emptyset$ except at the point p.

3.3.1 Finding the Minimum Vertex Distance between Two Separable Convex n-gons in the L_1 Metric

Given two separable convex n-gons, the Cartesian coordinates of the extreme vertices can be found in $O(\log n)$ by drawing lines parallel to the x or y axis, which pass these extreme vertices. The lines, say Y_1,Y_2,X_1,X_2, partition the

boundaries of the polygons into at most 10 *part chains* since the boundary of a convex polygon is unimodal with respect to its Cartesian coordinates. The cases are shown in Fig. 21.

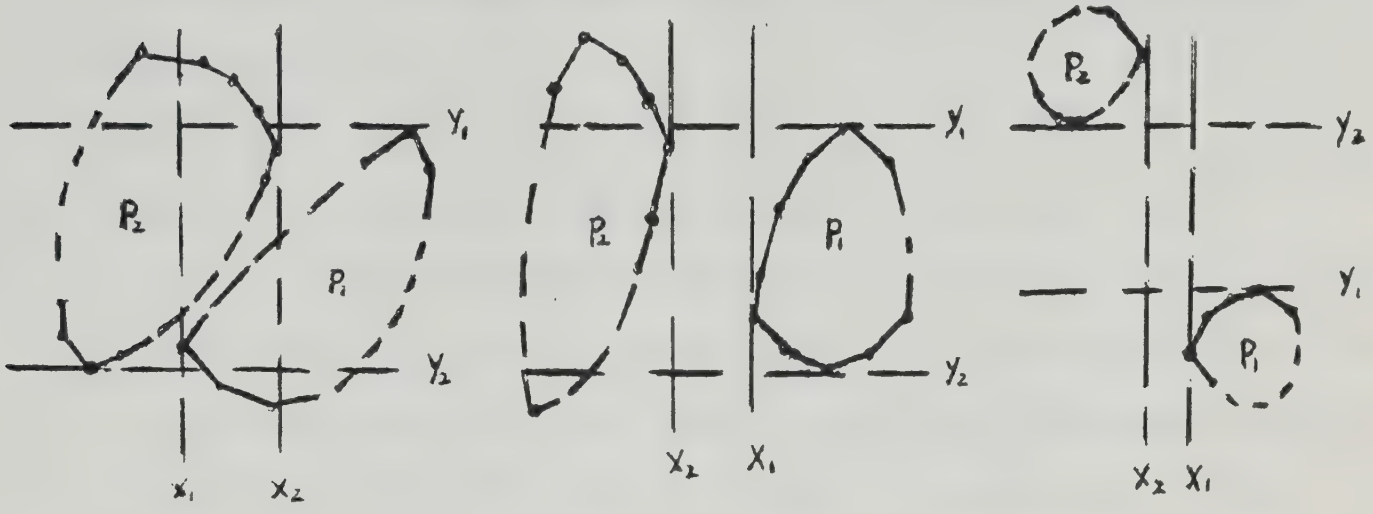


Fig. 21

Finding the minimum vertex distance of any two part chains between two polygons takes time linear with respect to the number of total vertices in these two part chains. Let y' be the intersection point of parallel lines $Y's$ ($X's$) and the boundary of polygons, possibly $y' \epsilon V_1(V_2)$. Let $S_1=\langle x,y\rangle$ denote a vertex part chain, where y is outside the chain $\langle x,y'\rangle$ and is the closest vertex to the point y'. The different cases are shown in Fig. 22.

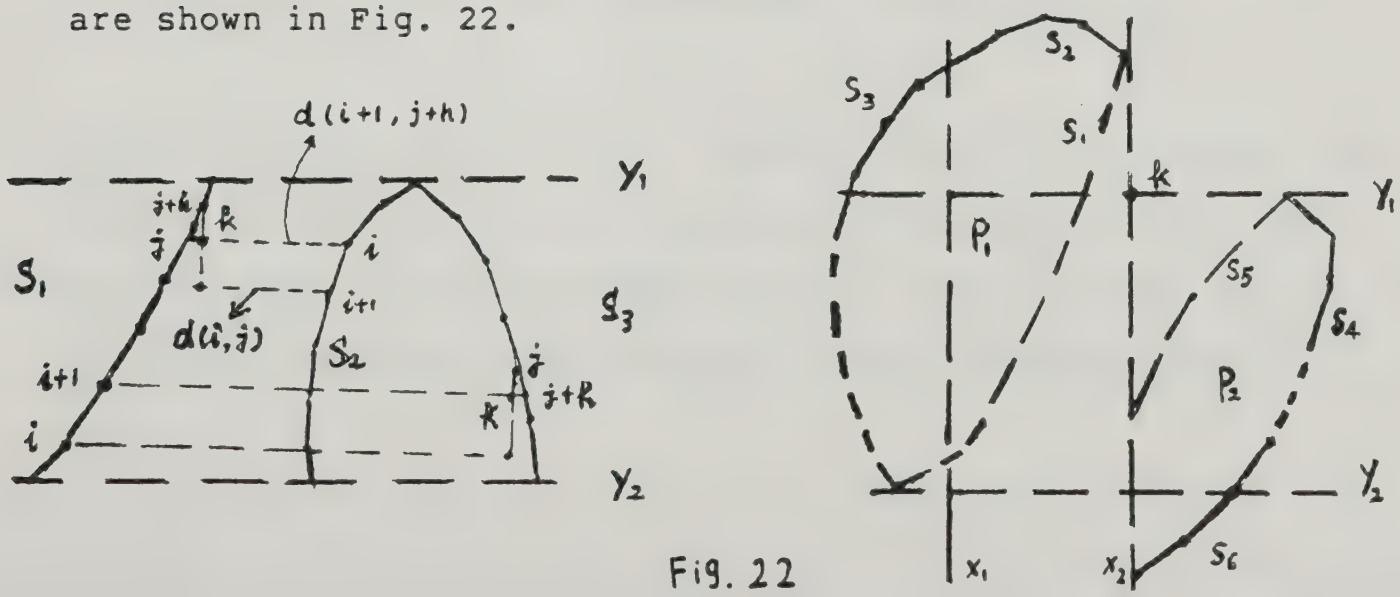


Fig. 22

Lemma 12: *The vertex part chains S_1, S_2, or S_1, S_3, which are between lines Y_1 and Y_2, are monotonic with respect to the mapping ω which maps each vertex of S_2 (S_3) into the closest vertex of S_1, or maps each vertex of S_1 into the closest vertex of $S_3(S_2)$.*

Proof:

1. Let $d(i,j)=d(i,S_1)$ and $d(i+1,j+h)=d(i+1,S_1)$ be two consecutive minimum vertex distances, where $i, i+1 \in S_2$ and $j, j+h \in S_1$. Assume that they intersect in k, where intersection means the endpoints of two distance swap as shown in Fig. 22. Since $d(i,j)$ is the minimum vertex distance, then $(k,j+h)>(k,j)$. This is contradicting to the assumption that $d(i+1,j+h)$ is the minimum vertex distance. Therefore, $d(i+1,j+h))$ and $d(i,j)$ do not intersect each other.
2. The coordinates of vertices in S_1, S_2, S_3 are monotonically decreasing (increasing) with respect to the y coordinates due to the convexity of the polygon. Thus, checking the minimum vertex distance along the vertex part chains need never backtrack; i.e., all mappings can be found in the time linear to the total number of the vertices in the two chains due to the convexity of $S's$. □

Lemma 13: *Given that S_6, S_5, S_4 and S_3, S_2, S_1 are not inside the same line pair X_1, X_2 or Y_1, Y_2, the minimum distance of any pair between the vertex chains S_6, S_5, S_4 and S_3, S_2, S_1 can be found in time linear with respect to the number of total*

vertices in the pair of chains.

Proof: Omitted, since it is enough to find the minimum distance only from k to $S's$, where k is the intersection point of $X's$ and $Y's$. And the minimum distance from k to $S's$ can be found in linear time. Fig. 22 illustrates this case.

Lemmas 12 and 13 exhaust all possible cases with two boundaries. In each case, the minimum vertex distance along the vertex part chain is checked without backtracking. As a result, an algorithm to compute the minimum vertex distances with time linear to the total number of vertices in two polygons can be designed. Finally, the following two theorems hold.

Theorem 12: *The minimum vertex distance in the L_1 metric between two separable n-gons can be found in $O(n)$ time.*

Theorem 14: *Finding the minimum vertex distance in the L_1 metric between two separable n-gons takes at least $O(n)$ time.*

3.3.2 Finding the Minimum Distance between two Separable Convex n-gons in the L_1 Metric

Consider the minimum distance between two boundaries of n-gons in the L_1 metric. The method for solving this problem is similar to that given in Section 3.2. Therefore, the proofs which are obvious are omitted. One can prove that if c_1 (resp. c_2) is the closest point of P_1 (resp. P_2), then c_1

(resp. c_2) is always visible from the extreme points in P_2 (resp. P_1). Hence, finding the minimum distance between two convex n-gons is reduced to finding the minimum distance between two visible chains. Let $V_1=V(P_1, S_2)$ (resp. V_2) be a visible chain, where S_2 (resp. S_1) is a set of extreme points in the x and y axes. The visible chain can be found in $O(\log n)$ by using a binary search on P_1, P_2 (i.e., determine the endpoints of visible chains).

The unimodel property of minimum distance between two visible chains also holds due to the convexity of visible chains V_1, V_2. Therefore, a solution to the remaining problem is shown by following lemma.

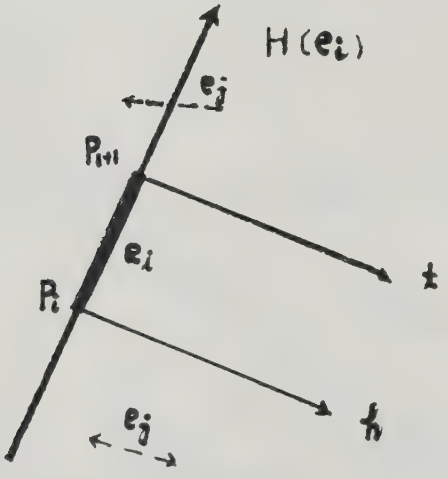


Fig. 23

Lemma 14: *Given a pair of edges, $(e_i=(p_i, p_{i+1}, e_j=(q_j, q_{j+1}))$ in V_1 and V_2, respectively, the result of either of the following*

two comparisons can be determined in constant time, (1) $\Delta_2(p_i)$ *vs* $\Delta_2(p_{i+1})$ *(2)* $\Delta_1(q_j)$ *vs* $\Delta_1(q_{j+1})$, *where* $e_i \epsilon P_1$, $e_j \epsilon P_2$

Proof: The proof method is the same as lemma 7.

By the symmetry of the two edges. Here only e_i is considered. Let $H(e_i)$ represent the half space formed by the infinite *extended edge* e_i.

1. If $H(e_i) \cap e_j \neq \emptyset$ as shown in Fig. 23, then by the convexity of P_1 and P_2, in the case of e_j above the ray t, where t is perpendicular to e_i originating at p_{i+1}, then $\Delta_2(p_i) > \Delta_2(p_{i+1})$; In the case of e_j below the ray h, where h is perpendicular to e_i originating at p_i, then $\Delta_1(p_{i+1}) > \Delta_1(p_i)$, by the convexity of P_2. (note that e_j can not lie between rays t and h because of visibility)

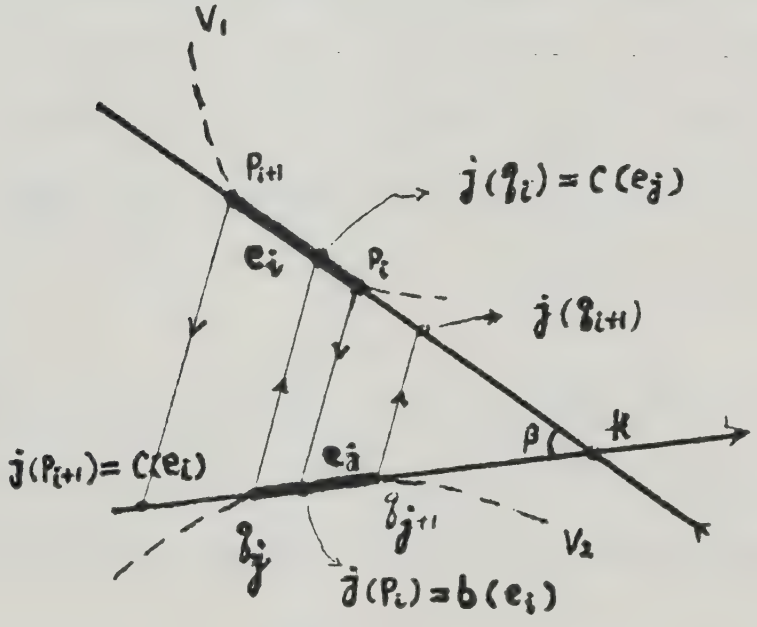


Fig. 24

2. If $H(e_i) \cap e_j = \emptyset$ *and* $H(e_j) \cap e_i = \emptyset$, then originating at the endpoints of edges p_i, p_{i+1}, and q_j, q_{j+1}, draw a set of

rays parallel to either the x or the y axis. These will be called *extended rays*. The extended rays and the extended edges intersect each other. Let $(k,c(e_i))=max\{(k,j(p_i)),(k,j(p_{i+1}))\}$ and $(k,b(e_i))=min\{(k,j(p_i)),(k,j(p_{i+1}))\}$, and $j(p_i)$, $j(p_{i+1})$ be the intersection points of extended edge e_j and extended ray e_i such that $j(p_i)$, $j(p_{i+1})$ are the closer pair to K, similarly for $c(e_j)$ and $b(e_j)$, and let k be the intersection point of the two extended edges, β be the intersection angle, as shown in Fig. 24. If $b(e_i)\subset e_j$ or $e_j\subset(k,b(e_i))$, then by the convexity of P_2 $d(p_{i+1},V_2)>d(p_i,V_2)$. If $e_j\subset(c(e_i),b(e_i))$, then due to the convexity of P_1 and P_2, $d(q_j,V_1)>d(q_{j+1},V_1)$. (refer to Fig. 24) If e_j is outside $(k,b(e_i))$ and e_i is outside $(k,b(e_j))$, then either $d(q_j,V_1)>d(q_{j+1},V_1)$ or $d(p_{i+1},V_2)>d(p_i,V_2)$. Note that either e_i, or e_j must be one of the above cases, and at least one of these two comparisons can be decided. □

In any case, one can determine the shorter distance. Therefore, an algorithm to find the minimum distance in $O(\log n)$ time can be designed. Finally, the following theorem holds.

Theorem 15: *Given two separable convex n-gons P_1, P_2, there exists an optimal algorithm which can find the minimum*

distance between them in O(log n) time.

3.3.3 Finding the Minimum Distance between a Convex n-gon and a Non-convex m-gon in the L_1 Metric

In order to achieve the optimal linear time bound, the boundary of Q is preprocessed such that the preprocessed $B(Q')$ and $B(P)$ have the monotonic property. The preprocessing for Q is similar to that in Section 4.1, and the steps are described as follows:

1. Find the extreme vertices of P in x,y coordinates, say (x_1,y_l), (x_2,y_r), (x_b,y_1), (x_t,y_2). At the extreme vertices, draw lines paralleling either x or y axis. These lines partition the plane into at most 12 regions, for each region, assign a pair of deques, as shown in Fig. 25.

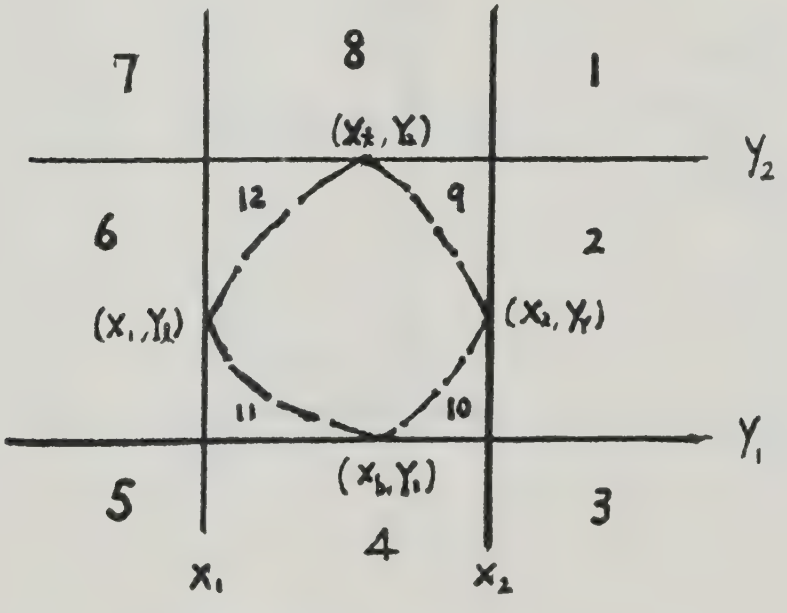


Fig. 25

2. Start from q_0, q_1, find the regions they locate;

connect these two vertices, and then find the intersection points with the four lines mentioned above. If these two vertices lie in different regions, push the line segments into their corresponding deques.

3. For an arbitrary vertex q_i, find the region it locates, connect (q_{i-1}, q_i), and find the points intersecting the four lines. For each line segment decide its visibility from its corresponding chain of P. Repeat step 3 until all vertices of Q are examined. In order to decide the visibility of a line segment, consider the following three subcases.

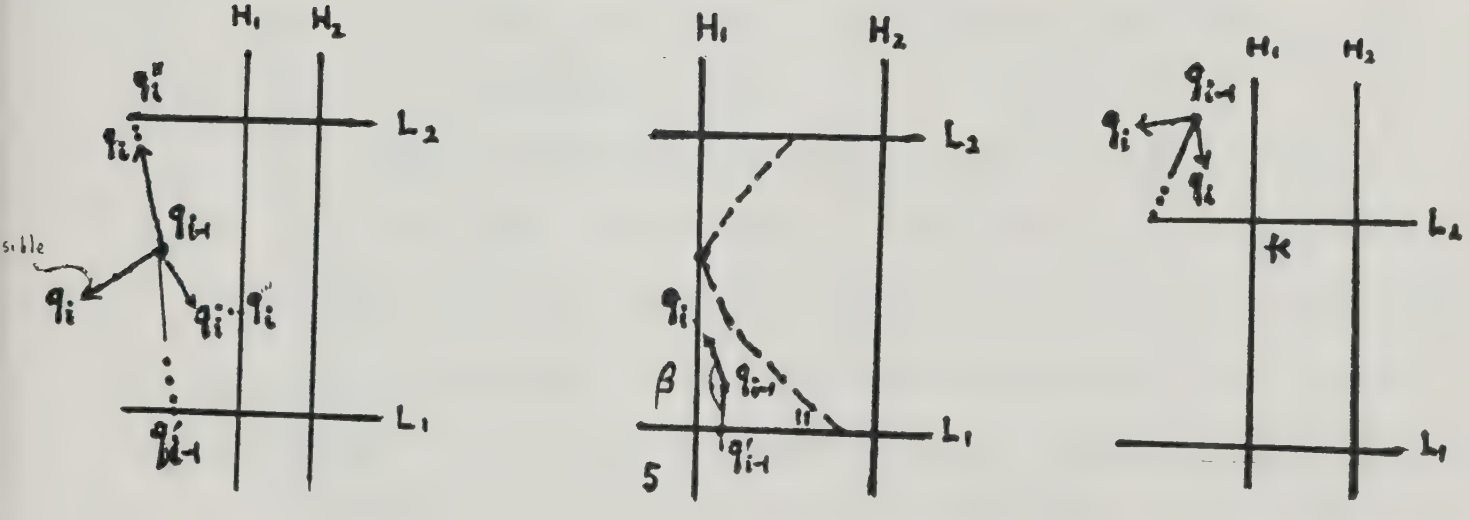


Fig. 26

a. Refer to Fig. 26a. If q becomes visible, then push q''_i or q_i into its deques and delete the part of the chain which previously belongs to the visible chain but now becomes invisible due to the new line segment (q_{i+1}, q_i). Otherwise, delete q_i or q''_i.

b. Refer to Fig. 26b. If q becomes invisible, i.e., $\beta \leq 0°$, then delete it. Otherwise, push it into the deques which are assigned to this region. If the two endpoints in the deques belong to the lines H_1 and L_2, respectively, then block the deques assigned to the region 5, because no vertices in region 5 will be closer than that in region 11 to the chain of P.

c. Refer to Fig. 26c. If q becomes visible from k, then push it into the deques, and delete the invisible part.

Each vertex of $B(Q)$ has at most 4 new vertices created, and each vertex is checked no more than twice. Hence the preprocessing takes only $O(m)$ time. Because the preprocessed $B(Q')$ and $B(P)$ are monotonic with respect to either the x or the y coordinates, therefore they are also monotonic with respect to the mapping ω, which maps each vertex of $B(Q')$ into its closest point in $B(P)$, and vice versa. The following two lemmas hold.

Lemma 15: *The preprocessed $B(Q')$ can be constructed in*

O(m) time, B(Q′) and B(P) are monotonic with respect to the mapping ω, which maps each vertex of B(Q′) into its closest point in B(P).

Lemma 16: *The minimum distance between a convex n-gon and a non-convex m-gon in the L_1 metric can be found in O(n) time.*

The vertex distance between them can also be found in O(n) time because the minimum vertex distances between them have the same property. Finally, the following theorem holds.

Theorem 16: *The minimum (vertex) distance between a convex n-gon and a non-convex n-gon in the L_1 metric can be found in O(n)* time.

If a convex *n*-gon entirely contains another convex *n*-gon, the minimum vertex distance in the L_1 metric can be found in *O(n)* time. The method is as follows: find the extreme vertices; draw the lines parallel to the x and y axes; partition two polygons into at most 12 part chains. Since these part chains have the monotonic property with respect to the x and y axes, it is easy to design an algorithm with time linear with respect to the number of vertices in the two polygons.

For the minimum vertex distance between a convex polygon and an included non-convex polygon in the metric

L_1, such that the closest vertices must be visible each other, whether or not a linear time algorithm can be designed is unsolved.

3.3.4 Minimum Distance between two Separable Non-convex Simple n-gons

There currently exists no sub-quadratic algorithm for solving the minimum distance problem between two separable non-convex simple polygons. In this section, an $O(n \log m)$ algorithm to solve this problem is presented, where m is the number of cusp vertices [Preparata 1977] in the polygons. The algorithm is based on the following facts: the minimum distances from the vertices in one non-convex polygon to another separable non-convex polygon are multimodal; The vertex must be a cusp vertex and has 90° degree visible angle; If the boundary of two non-convex polygons has been preprocessed, then these two preprocessed boundaries have *pseudo-monotonic property* with respect to a mapping ω', which maps each vertex of one boundary to its closest point in a certain part of another boundary (Pseudo-monotonic means that the mapping ω' maps a vertex in preprocessed boundary of Q_1 (or Q_2) to the local closest point in preprocessed boundary of Q_2 (or Q_1) instead of mapping a vertex of Q_1 (or Q_2) to the global closest point in preprocessed boundary of Q_2 (or Q_1) without loss of the minimum distance between two Q's, where local closest point is a closest point from a vertex to a certain part of boundary bounded by two consecutive closest

points or its 90° degree visible angle while global closest point is a closest point from a vertex to whole boundary); Lemma 18 will show that if each vertex is mapped to its local closest point, the minimum distance between two polygons can not be missed; Since the endpoints of the minimum distance between the two polygons must be on the preprocessed boundaries, then check the preprocessed boundaries is adequate; By using this pseudo-monotonic property, one can apply a divide-conquer method on the two preprocessed boundaries to find the minimum distance between them.

In the following, a preprocessing procedure is first described, then two lemmas are proved, finally the algorithm is briefly discussed.

The preprocessing procedure is as follows;

1. Find the convex hull of Q_1,Q_2, and draw a separating line L between the two convex hulls.
2. For each vetex of Q_1 (respectively, Q_2) draw a line perpendicular to L and intersecting L at a point. If the perpendicular line does not intersect Q_1 at another point, the vertex of Q_1 is called *perpendicular visible* from L. The perpendicular visible chain of Q_1 (Q_2) from L can be easily decided (Refer to Fig. 27a).
3. Construct a vertex chain based on the perpendicular visible chain such that each vertex in the new chain is cusp vertex[Preparata 1977] (Refer to Fig. 27b).
4. Construct a new cusp vertex chain after step 2 such that

each cusp vertex has 90° degree visible angle by scanning twice of the cusp chain.

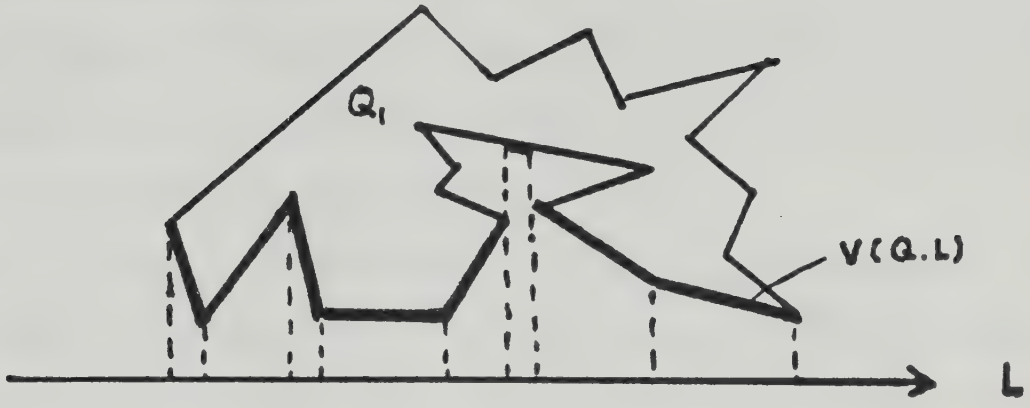


Fig. 27

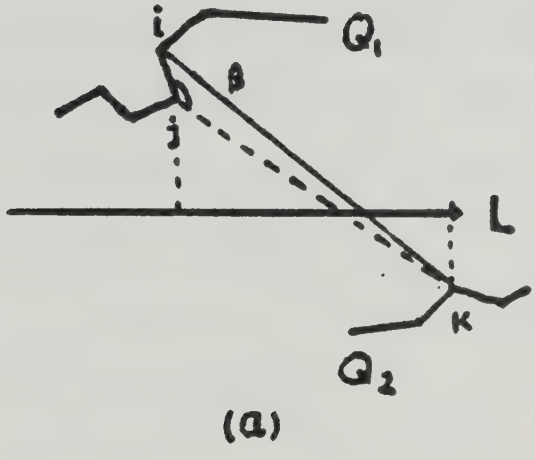
Lemma 17: *The minimum distance between two separable simple non-convex polygons is equal to the minimum distance between two preprocessed boundaries which can be constructed in $O(n)$ time.*

Proof: (refer to Fig. 28) Assume i is not perpendicular invisible from L, and $d(Q_1,Q_2)=(i,k)$. Clearly, (j,k) is less than (i,k) since $\beta>90°$, contradicting the assumption. Both step 1, 2 and 3 take $O(n)$ time. Step 4 takes $O(m)$ since the cusp chains are monotonic with respect to a mapping, which maps each vertex in the boundaries to its perpendicular point in L and the procedure equals to trianglate a monotone simple polygon [Garey M. etc. 1978]. Therefore, the preprocessing takes only $O(n)$.□

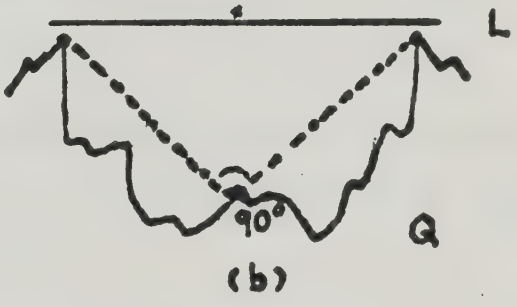
Lemma 18: *The preprocessed boundary of* Q_1 (Q_2) is monotonic with respect to a mapping ω', which maps each vertex in the preprocessed boundary of Q_1 to its local closest point in the preprocessed boundary of Q_2.

Proof: Case 1. Refer to the Fig. 29. If i is bounded by two consecutive colosest pairs (k,m) and (k',m'), then (k,j) or (i,m) is shorter than (i,j).

Case 2. Refer to the Fig. 30. If i is bounded by two neighbor vertices which form 90° degree visible angle of i, , then (m',k') is shorter than (m',k) or (j,k) is shorter than (m,k).□



(a)



(b)

Fig. 28

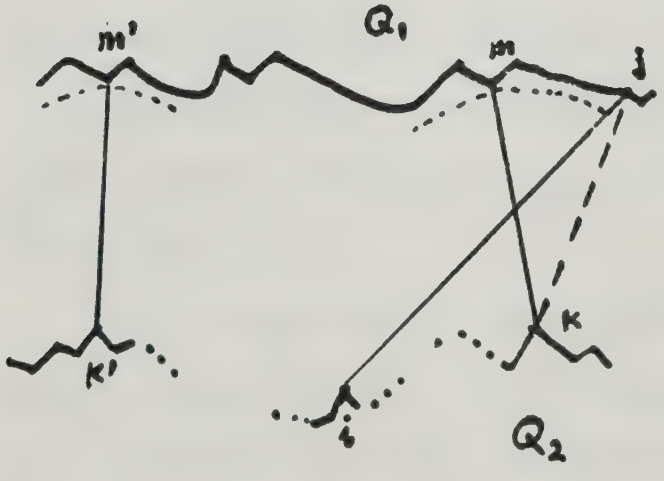


Fig. 29

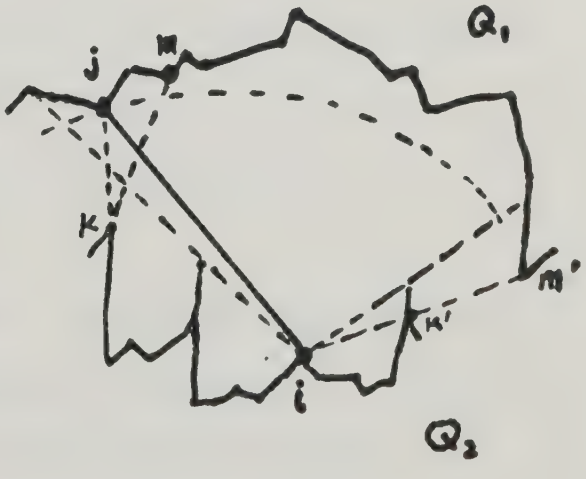


Fig. 30

Chapter 4

Conclusion

By investigating three fundamental polygon problems, this thesis provides some evidence to show that efficient algorithms for three problems can be derived if one takes advantage of the monotonic and convex properties of polygons. The monotonic property plays an important role in some non-convex object problems. As a result, this may be a simple way to solve some non-convex polygon problems without using *VORONOI* diagrams. Monotonicity can be used to find the diameter of a convex polygon (hence, a non-convex polygon by constructing its convex hull) or to find the nearest neighbors of a convex polygon, both in $O(n)$ time, which are the same time complexity as those of Lee and Praparata, [1977] and Shamos, [1975].

Extending the intersection problem to higher dimensions is very important to linear programming. It is still an open problem.

Lower bounds have great theoretical significance. The oracle method [Knuth 1973] is applied on the proofs of several lower bounds shown in this thesis. But other non-trivial lower bounds remain unsolved. They may require new techniques.

In the minimum distance problems, the visibility property plays an essential role in the design of algorithms. This is not unexpected in view of the close relationship beween classical optical physics phenomena and geometric principles.

The minimum distance in the L_1 metric has practical meaning if one considers phenomena in an optical anisotropic medium.

All results are summarized in the table.

Problems \ Objects		Two Simple Non-Convex N-gons	A Convex N-gon & A Non-Convex M-gon	Two Convex N-gons
Polygon Intersection Reporting Problem		O((n+k)log n)* (Bentley & Ottmann)	O(mlog n)*	O(n) (Shamos)
Polygon Intersection Detecting Problem		O(nlog n) (Shamos)	O(n) O(n) (R(Q)⊂R(P))	O(log n) (Chazelle & Dobkin) O(n) (R(P_1)⊂R(P_2))
Minimum Distance Problem	Euclidean (L_2) Metric	O(nlog m)*	O(n) (boundary) O(n) (R(Q)⊂R(P)) O(n) (vertex)	O(log n) (boundary) O(n) (R(P_1)⊂R(P_2) & Vertex or R(P_1)⊂R(P_2) & boundary) O(n) (vertex)
	(L_1) Metric		O(n) (boundary) O(n) (vertex)	O(log n) (boundary) O(n) (R(P_1)⊂R(P_2) & vertex or R(P_1)⊂R(P_2) & boundary) O(n) (vertex)

Note: * Means the algorithm is not optimal or it has not been proved to be optimal.

References

1. Aho A., Hopcroft J., Ullman J., *The design and analysis of computer algorithms*, Addison-Wesley Publishing Company, Reading, Massachussetts, 1974.

2. Akl S., Toussaint G., 'A fast convex hull algorithm,' *Information Processing Letters*, pp. 216-219, 1979.

3. Appel A., 'Determining the three dimensional convex hull of a polyhedron,' *IBM Journal of Research and Development*, pp. 590-601, 1976.

4. Avis D., Toussaint G., 'An optimal algorithm for determining the visibility of a polygon from an edge,' *IEEE Transactions on Computers*, pp. 910-1014, 1981.

5. Bentley J., 'Multidimensional divide and conquer,' *Communications of the ACM*, pp. 214-229, 1980.

6. Bentley J., Ottmann T., 'Algorithms for reporting and counting geometric intersections,' *IEEE Transactions on Computers*, pp. 643-647, 1979.

7. Boyce J., Dobkin D., 'Finding extremal polygons,' *Journal of the ACM*, pp. 282-289, 1982.

8. Brown K., 'Geometric transforms for fast geometric algorithms,' *PhD Thesis*, Carnegie-Mellon University, Dept. of Computer Science, 1979.

9. Bykat, A., 'Convex hull of a finite set of points in two dimensions,' *Information Processing Letters* pp. 296-298, 1978.

10. Chazelle B., Dobkin D., 'Detection is easier than

computation,' *ACM Symposium on Theory of Computing*, pp. 146-153, 1980.

11. Chazelle B., Dobkin D., 'Decomposing a polygon into its convex parts,' *Procedings of the 11th ACM SIGACT Symposium*, Los Angeles, California, pp. 38-48, 1980.
12. Chin F., Wang C., 'Optimal algorithms for the intersection and the minimum distance problems between planar polygons,' to appear in *IEEE Transactions on Computers*, 1983.
13. Chin F., Wang C., 'Minimum vertex distance between two separable convex polygons,' to appear in *Information Processing Letters*, 1983.
14. Dobkin D., Lipton R., Reiss S., 'Excursions into geometry,' *Technical Report* No.71, Dept. of Computer Science, Yale University, 1976.
15. Dobkin D., Kirkpatrick D., 'Fast detection of polyhedral intersection,' To appear.
16. Edelsbrunner H., Maurer H., 'Polygonal intersection searching,' *Information Processing Letters*, pp.74-79, 1982.
17. Fisher M., 'Fast algorithms for two maximal distance problems with applications to image analysis,' *Pattern Recognition*, pp. 35-40, 1975
18. Freeman H., Shapira R., 'Determining the minimum-area incasing rectangle for an arbitrary closed curve,' *Communications of the ACM*, pp. 409-413, 1975.
19. Friedman J., Basket F., Shustek L., 'An algorithm for

finding nearest neighbors,' *IEEE Transactions on Computers*, pp. 1001-1006, 1975.

20. Garey M., Johson D., Preparata F., Tarjan R., 'Trianglating a simple polygon,' *Information Processing Letters*, pp. 175-179, 1978.
21. Gindy H., Avis D., ' A linear algorithm for determining the visibility polygon from a point,' *Journal of Algorithms*, pp. 186-197, 1981.
22. Grossman D., 'Prodedural representation of three dimensional objects,' *IBM Journal of Research and Development*, pp. 582-588, 1976.
23. Kirkpatrick D., 'Optimal search in planar subdivisions,' *SIAM Journal on Computing*, 1982.
24. Knuth D., *'The art of computer programming,' Vol.3*, Addison-Wesley, 1973.
25. Lee D., 'Two dimensioal Voronoi Diagrams in Lp metric,' *Journal of the ACM*, pp.604-618, 1980.
26. Lee D., Preparata F., 'An optimal algorithm for finding the kernel of a polygon,' *Journal of the ACM* , pp. 415-421, 1979.
27. Muller D., Preparata F., 'Finding the intersection of two convex polyhedra,' *Technical report*, University of Illinois, 1977.
28. Preparata F., 'Steps into computational geometry I and II,' *Technical report*, University of Illinois, 1977.
29. Reif J., 'Complexity of the mover's problem and generalizations extended abstract,' *Proceedings of the*

20th Annual IEEE Symposium on Foundations of Computer Science, pp. 421-427, 1979.

30. Schachter B., 'Decomposition of polygons into convex sets,'*IEEE Transactions on Computers*,, pp. 365-374, 1978.
31. Shamos M., 'Geometric complexity,' *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pp. 224-233, 1975.
32. Shamos M., Hoey D., 'Geometric intersection problems,' *Proc. 17th Annual Conf. Foundations of Computer Science*, pp. 208-215, 1976.
33. Schwartz J., 'Finding the minimum distance between two convex plygons,' *Information Processing Letters*, pp. 168-170, 1981.
34. Snyder W., Tang E., 'Finding the extrema of a region,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 266-269, 1980.
35. Toussaint G., 'Pattern recongnition and geometrical complexity,' *5th International Conference on Pattern Recognition*, pp. 1324-1347, 1980.
36. Vaishnavi V., Kriegel H., Wood D., 'Space and time optimal algorithms for a class of rectangle intersection problems,' *Information Sciences* 21, pp. 59-67, 1980.
37. Willard D., 'Polygon retrieval,' *SIAM, Journal of Computing*, pp. 149-165, 1982.